Universitat Oberta de Catalunya

uoc.edu

Astronomic Timer

Miguel Ángel García Pérez
Desarrollo de Aplicaciones para Dispositivos Móviles
Trabajo final de máster DADM aula 4

Carles Garrigues Olivella
David Escuer Latorre

Fecha Entrega

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Astronomic Timer*
Nombre del autor:	*Miguel Ángel García Pérez*
Nombre del consultor/a:	*David Escuer Latorre*
Nombre del PRA:	*Carles Garrigues Olivella*
Fecha de entrega (mm/aaaa):	MM/AAAA
Titulación:	*Máster: Desarrollo de Aplicaciones para Dispositivos Móviles*
Área del Trabajo Final:	*Trabajo final de máster DADM aula 4*
Idioma del trabajo:	*Español*
Palabras clave	*Aplicación* *Android*

Resumen del Trabajo (máximo 250 palabras): *Con la finalidad, contexto de aplicación, metodología, resultados y conclusiones del trabajo.*

El fin de este proyecto es poder poner en práctica los conocimientos de diseño y desarrollo entre otros, adquiridos durante la realización del máster para la creación de aplicación nativa para Android.

La aplicación nace de una serie de requisitos técnicos que desde un principio queríamos implementar en el proyecto como son la obtención de datos de un servidor, la persistencia de datos en base de datos del dispositivo, uso de fragments, mapas, etc. Para implementar estos requisitos decidimos el contexto de la astronomía y planteamos los requisitos para adaptarlo.

Para llevar a cabo todas las fases de este proyecto optamos por una metodología *agile* que nos diera la flexibilidad necesaria ejecutar un proyecto de estas características donde una persona ejecuta e implementa todas las fases del proyecto.

Como resultado hemos obtenido una aplicación totalmente funcional que ha sido posible gracias a una gestión adecuada de los tiempos ciñéndonos en la mejor medida posible a la planificación acordada, pero también al trabajo realizado durante la fase de diseño donde se ideo y dio forma a todos los requisitos que conformarían la aplicación final.

Abstract (in English, 250 words or less):

The aim of this project is to be able to put into practice the knowledge of design and development among others, acquired during the realization of the master for the creation of native application for Android.

The application was born from a series of technical requirements that we wanted to implement in the project, such as obtaining data from a server, persisting data in the device's database, using fragments, maps, etc. To implement these requirements, we decided the astronomy context and set out the requirements to adapt it.

To carry out all the phases of this project we chose an agile methodology that would give us the necessary flexibility to execute a project of these characteristics where a person executes and implements all the phases of the project.

As a result we have obtained a fully functional application that has been possible thanks to an adequate management of the times, adhering in the best possible way to the agreed planning, but also to the work carried out during the design phase where all the designs were designed and shaped. requirements that would make up the final application.

Índice

Lista de tablas

Lista de figuras

1. Introducción

1.1 Contexto y justificación del Trabajo

Hoy día, una parte muy relevante de la sociedad está acostumbrada a consumir, bien en exclusiva o, muy frecuentemente información cotidiana en formato digital. Esta conducta es, en gran medida, la derivada de la implantación de los dispositivos móviles en la vida de los usuarios que, gracias a las conexiones inalámbricas de alta velocidad, pueden consumir al instante cualquier tipo de información en prácticamente cualquier lugar vía internet. Gracias a la gran aceptación de estos dispositivos entre la sociedad, a su facilidad de uso y a que cada vez son más potentes y completos la oferta de aplicativos no para de crecer.

Con este contexto es fácilmente entendible que se pueda plantear prácticamente cualquier solución que facilite en algún aspecto la vida del usuario final. Y no sería un error presuponer que una solución en soporte móvil para una necesidad detectada pueda tener éxito con la difusión adecuada.

Este proyecto pretende poner a disposición del usuario información relevante acerca de diferentes tipos de fenómenos astronómicos.
La necesidad de implementar esta solución se detecta cuando, bien por las noticias, u otro medio, se hacen eco de que próximamente se va a producir un fenómeno astronómico importante, se puede observar que causa interés entre una parte importante de la sociedad. Pero este ámbito de difusión está limitado a que el usuario este consumiendo noticias en un horario y medio determinado, o que proactivamente se interese en buscar esta información. Estas limitaciones hacen que no llegue a mucha gente y aquí es donde radica la necesidad de crear un medio por el cual un usuario pueda estar al tanto de los próximos eventos de su interés con tan solo consultar su smartphone.
La solución planteada en este proyecto cubrirá la necesidad tanto de usuarios aficionado a la astronomía como de usuarios “curiosos” que puntualmente pudieran estar interesados en determinado fenómeno.

En el mercado de aplicaciones actualmente ya existen diferentes apps que intentan resolver esta misma necesidad, pero cada una de ellas da un enfoque diferente a la solución final y la forma de presentarlo. Muchas de ellas estás especializadas en un solo tipo de fenómeno y por lo general tienen gran cantidad de información acerca de los mismo, por lo que no es un aspecto en el que, a priori, nos interese competir. El análisis de la competencia puede darnos información valiosa sobre las debilidades que un usuario pueda ver en estas soluciones y aprovecharlo para incluirlas como puntos fuertes de nuestra aplicación.

En un primer ejercicio analizaremos varias aplicaciones disponibles en Google Play, y de ellas revisaremos sus puntos fuertes y también sus carencias.

	**Eclipse Calculator 2.0**
	https://play.google.com/store/apps/details?id=calcEclipsi2.src

Puntos fuertes:

Acceso a los datos de todos los eclipses de Sol y Luna y de los tránsitos planetarios entre los años 1900 y 2100
Cálculo de las circunstancias generales de cada fenómeno, incluyendo mapa de visibilidad.
Cálculo de las circunstancias locales para cualquier lugar del mundo (inicio, final, duración, altura del Sol y la Luna sobre el horizonte, ...).
Mapas interactivos para conocer las circunstancias del eclipse.
Simulación del fenómeno desde tu punto de observación.
Simulación del recorrido de la sombra de la Luna sobre la superficie de la Tierra (eclipses de sol).
Simulación de la trayectoria de la Luna a través de la sombra de la Tierra (eclipses de luna)
Puedes escoger el lugar de observación utilizando la base de datos incorporada, manualmente o a partir del GPS.
Cuenta atrás.
Multilingüe; Castellano, catalán e inglés.

Debilidades:

Es una aplicación muy completa para la necesidad que pretende cubrir, no podemos acentuar una debilidad reseñable en esta aplicación. Revisando las reseñas que han dejado en Google Play usuarios de esta app se confirma que cumple todas las expectativas.
Podemos tomar esta aplicación como referencia para definir algunos aspectos de nuestro proyecto dentro de la tipología de eclipses.

Tabla 1. Análisis aplicación Eclipse Calculator 2.0

	**Fase Lunar Pro**
	https://play.google.com/store/apps/details?id=com.daylightmap.moon.pro.android

Puntos fuertes:

Muestra instantáneamente la fase, el ángulo creciente, las horas de salida y puesta de luna y las sizigias más cercanas para cualquier posición y fecha
Super Luna, Luna Azul, Luna Negra y Eclipses lunares
Cambia la fase en la pantalla táctil o introduciendo una fecha
Arrastra con dos dedos para ver la cara oculta
El calendario muestra las fases del mes en un solo vistazo
El panel de datos ofrece datos adicionales: distancia, acimut, altitud, tránsito, apogeo, perigeo y más
Totalmente adaptado para tabletas
Notificaciones para las grandes fases
Fondos de pantalla animados

Debilidades:

Opciones no disponibles dependiendo del modelo de terminal.
Únicamente muestra información ordenada por meses, no permite el filtrado otro criterio.
Tiene un coste obligado de 0,99

Tabla 2. Análisis aplicación Fase Lunar Pro

	**Meteor Shower Calendar**
	https://play.google.com/store/apps/details?id=Bcom.ccwilcox.meteorshower

Puntos fuertes:

Tiene sistema de notificaciones de próximas lluvias de estrellas.
Widget para el icono de la pantalla principal para mostrar el número de días hasta la próxima lluvia de estrellas
También incluye información sobre horas de salida/puesta del sol desde tu ubicación, y la fase actual de la luna
Acceso al detalle de próximos eventos de lluvia de estrellas.
Ofrece la opción de adaptar el tema de la app para modo noche o día automáticamente.
Permite llevar a cabo búsquedas de Google dentro de la aplicación para leer más acerca de cada lluvia de estrellas.
Permite compartir con amigos a través de SMS, correo electrónico, Facebook, Google, Twitter, etc.
Ofrece pronóstico de tiempo para la fecha del próximo evento

Debilidades:

Incluye publicidad
Únicamente disponible en inglés
No adaptada a tablets.
Diseño y experiencia de usuario muy pobre.
No indica los puntos geográficos desde donde se puede apreciar la lluvia.
No permite personalizar las notificaciones.

Tabla 3. Análisis aplicación Meteor Shower Calendar

Analizando el mercado de aplicaciones que dan soluciones dentro de la misma temática que la aplicación que desarrollamos en este proyecto podemos observar que nuestro principal punto diferencial será que va a englobar en la misma aplicación más tipos de fenómenos diferentes que ninguna otra. Por lo general las aplicaciones analizadas se centran en un tipo de fenómeno astronómico concreto, lo que en realidad es un punto fuerte según están pensada esas aplicaciones, que es profundizar mucho sobre esos eventos con todo tipo de información de interés. Pero para el enfoque que pretendemos en este proyecto es un punto diferencial el abarcar varios tipos de fenómenos diferentes ya que no vamos a proporcionar información en detalle de los eventos más allá de las fechas y tipos. Esto nos permitirá llegar a un número mayor de usuarios que se sientan atraídos por unos u otros fenómenos.

Un punto fuerte y diferencial en nuestra aplicación debería ser la posibilidad de programar alertas de una forma bastante flexible, permitiendo al usuario indicar la antelación con la que quiere recibir el aviso, en número de repeticiones, posibilidad de agentar en el calendario una cita. Toda esta funcionalidad combinada deberá ser el núcleo de la aplicación, poder consultar y ofrecer avisos configurables de diferentes eventos astronómicos.

1.2 Objetivos del Trabajo

Requisitos Funcionales	
RF.TFM.001	Visualizar en todo momento un panel en la parte superior
RF.TFM.002	El usuario debe poder volver a la ventana inicial y al menú desde en cualquier momento
RF.TFM.003	Proporcionar al usuario un menú de navegación basado en pestañas
RF.TFM.004	La información de cada tipo de fenómeno debe estar en ventanas diferentes
RF.TFM.005	Permitir al usuario listar eclipses y lluvias de estrellas
RF.TFM.006	La presentación de las fases lunares ha de ser tipo calendario gregoriano
RF.TFM.007	Los listados de fenómenos deberán permitir el filtrado por fecha
RF.TFM.008	Presentar al usuario información detallada de un fenómeno seleccionado
RF.TFM.009	El usuario debe poder consultar detalles como el tiempo hasta un evento.
RF.TFM.010	Se podrá consulta un mapa de visibilidad de un eclipse
RF.TFM.011	Menú general de tipo desplegable con acceso a ajustes, alertas y ayuda
RF.TFM.012	Se deberá presentar por separado los ajustes de cada tipo de alerta
RF.TFM.013	La información de los fenómenos deberá recuperarse de un servidor web.

Tabla 4. Listado de requisitos funcionales

Requisitos No Funcionales	
RNF.TFM.001	Deberá poder ejecutarse en cualquier dispositivo Android
RNF.TFM.002	Solo se podrá visualizar en modo *portrait*
RNF.TFM.003	Deberá estar disponible en castellano e inglés
RNF.TFM.004	Deberá poder dar servicio sin conexión a internet
RNF.TFM.005	Deberá gestionar los permisos del dispositivo correctamente

Tabla 5. Listado de requisitos no funcióneles

1.3 Enfoque y método seguido

En desarrollo del proyecto vamos a seguir patrones de las metodologías *agiles*, como las entregas por *sprints*, la autogestión, la medición del progreso, incluir nuevos requisitos, etc. El aspecto que más nos interesa de estas metodologías es el poder hacer una planificación de hitos que se consigan tras cada sprint y donde se pueda ir entregando una solución completamente funcional, aunque no completa. Esto nos permitirá ir chequeando en fases tempranas del desarrollo posibles aspectos de mejora sobre lo definido e incluir nuevos requisitos.

“Por definición, las metodologías ágiles son aquellas que permiten adaptar la forma de trabajo a las condiciones del proyecto, consiguiendo flexibilidad e

inmediatez en la respuesta para amoldar el proyecto y su desarrollo a las circunstancias específicas del entorno."[1]

Los principios de las metodologías ágiles son 12, y se definieron en el Manifiesto Agile publicado en el 12 de febrero de 2001 y en el que participaron 17 autores. Estos principios pueden consultarse en Wikipedia[2]

Se ha elegido realizar este proyecto de forma nativa para Android, la decisión no ha sido técnica puesto que la aplicación según está planteada tendría sentido hacerla multiplataforma ya que las funcionalidades que implementa son simples y fácilmente replicables entre plataformas. La decisión es puramente práctica, mi intención principal al cursar este máster era especializarme en el desarrollo de aplicaciones nativas para Android e IOS, y de entre estas el decantarme por una u otra ha sido debido a que he trabajado menos el desarrollo para Android y con la ejecución de este proyecto me ayudará a profundizar y adquirir nuevos conocimientos.

1.4 Planificación del Trabajo

La planificación del trabajo a realizar durante el trabajo de fin de máster se detalla en el siguiente diagrama de Gantt.

Nombre de la tarea	Duración	Inicio	Finalizar	% Dedicación
Capítulo 1 de la Memoria	22d	20/02/19	13/03/19	
Contexto y justificación del Trabajo	4d	20/02/19	23/02/19	100%
Objetivos del Trabajo	3d	24/02/19	26/02/19	100%
Enfoque y método seguido	1d	27/02/19	27/02/19	100%
Planificación del Trabajo	7d	28/02/19	06/03/19	100%
Breve sumario de productos obtenidos	1d	07/03/19	07/03/19	100%
Breve descripción de los otros capítulos de la memoria	1d	08/03/19	08/03/19	100%
Repaso completo capítulo 1	5d	09/03/19	13/03/19	100%
Diseño	21d	14/03/19	03/04/19	
Usuarios y contexto de uso	1d	14/03/19	14/03/19	100%
Diseño conceptual	2d	15/03/19	16/03/19	100%
Prototipado	6d	17/03/19	22/03/19	65%
Evaluación	5d	18/03/19	22/03/19	35%
Definición de casos de uso	8d	23/03/19	30/03/19	
Diagrama UML	6d	23/03/19	28/03/19	50%
Listado casos de uso	7d	24/03/19	30/03/19	50%
Diseño de la arquitectura	9d	26/03/19	03/04/19	
Diagrama UML de BBDD	3d	26/03/19	28/03/19	100%
Diagrama UML entidades y clases	3d	29/03/19	31/03/19	100%

[1] Vanessa Rosselló Villán (03 OCT 2018) [en línea] Madrid. [Consulta 10 Mar 2019] [Disponible en: https://www.iebschool.com/blog/que-son-metodologias-agiles-agile-scrum/]
[2] Wikipedia (17 feb 2019) [en línea] [Consulta: 10 Mar 2019] [Disponible en: https://es.wikipedia.org/wiki/Manifiesto_%C3%A1gil]

Diagrama arquitectura de sistema	2d	02/04/19	03/04/19	100%
Implementación	42d	04/04/19	15/05/19	
Documentación en memoria	42d	04/04/19	15/05/19	10%
Desarrollo de servidor	5d	04/04/19	08/04/19	70%
Desarrollo de módulo principal	5d	09/04/19	13/04/19	70%
Desarrollo ventana Eclipse + detalle	8d	14/04/19	21/04/19	70%
Desarrollo ventana F. lunares + detalle	8d	22/04/19	29/04/19	70%
Desarrollo ventana Ll.estrellas + detalle	8d	30/04/19	07/05/19	70%
Desarrollo menú configuraciones	8d	07/05/19	14/05/19	70%
Pruebas	42d	04/04/19	15/05/19	10%
Correcciones sobre hitos anteriores	42d	04/04/19	15/05/19	10%
Entrega	21d	16/05/19	05/06/19	
Preparación entrega	21d	16/05/19	05/06/19	10%
Manual de usuario	5d	16/05/19	20/05/19	80%
Manual de instalación	5d	21/05/19	25/05/19	80%
Completar y revisar memoria	21d	16/05/19	05/06/19	10%
Presentación	11d	26/05/19	05/06/19	80%

Tabla 6. Diagrama de Gantt. Planificación

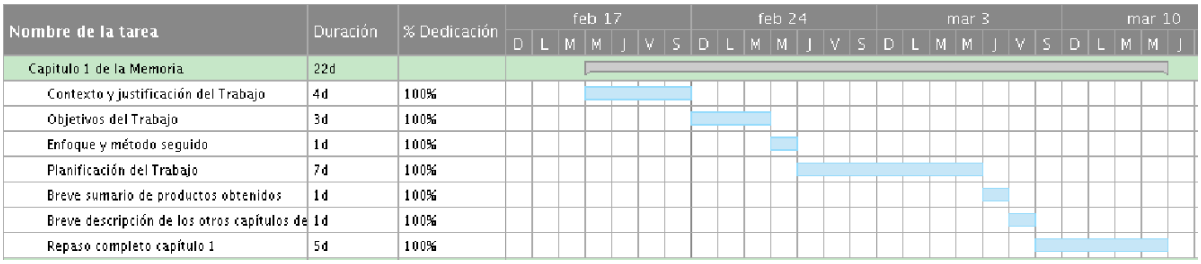


Figura 1. Diagrama de Gantt. Capítulo 1

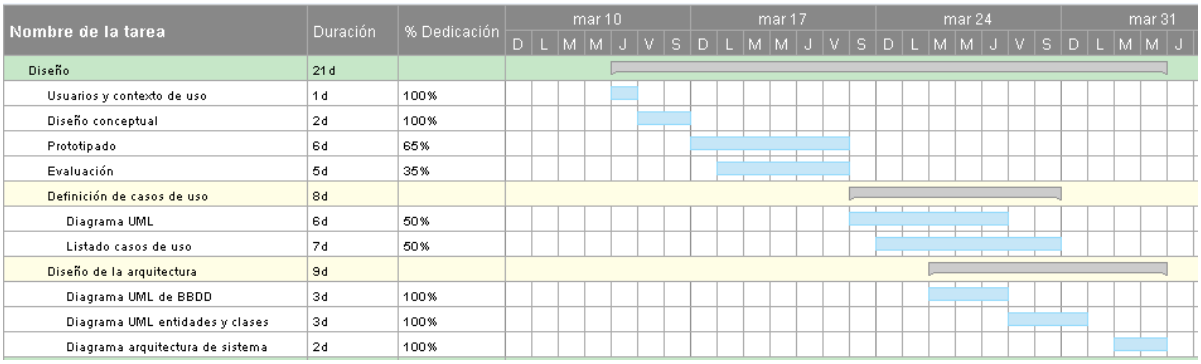


Figura 2. Diagrama de Gantt. Diseño

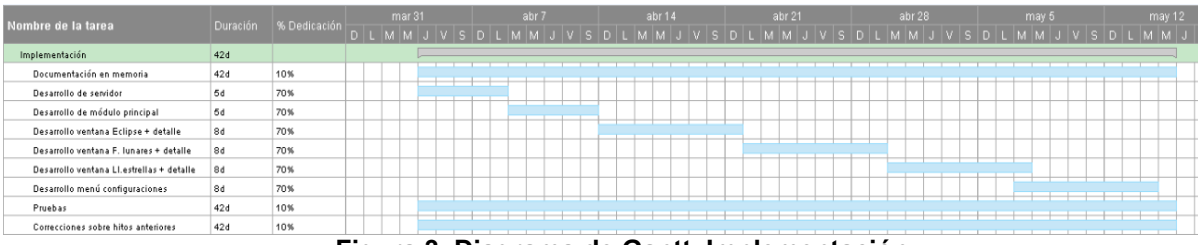


Figura 3. Diagrama de Gantt. Implementación

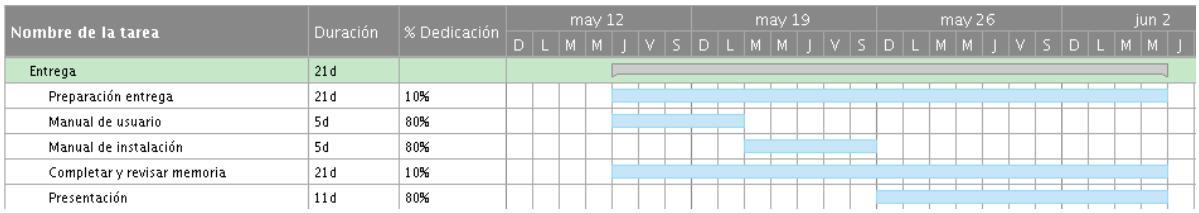


Figura 4. Diagrama de Gantt. Entrega

1.5 Breve sumario de productos obtenidos

- ✓ Memoria del proyecto en formato PDF
- ✓ Código fuente de la app, recursos y ejecutable apk en un fichero ZIP
- ✓ Manual de instalación en emulador en formato PDF
- ✓ Manual de instalación en dispositivo en formato PDF
- ✓ Prototipo del diseño del pruecto en formato .vp (JustinMind)
- ✓ Video de presentación del proyecto

1.6 Breve descripción de los otros capítulos de la memoria

- ➢ Capítulo 2: Dedicado al Diseño del proyecto
- ➢ Capítulo 3: Dedicado a la implementación del proyecto
- ➢ Capítulo 4: Dedicado a las conclusiones
- ➢ Capítulo 5: Recopilación de glosario de terminos
- ➢ Capítulo 6: Bibliografía
- ➢ Capítulo 7: Incluye los anexos de la memoría

2. Diseño

2.1. Usuarios y contexto de uso

Como punto de partida necesitamos identificar el perfil y las necesidades de nuestros usuarios potenciales. A continuación, describiremos un perfil tipo de usuario en una ficha de persona que nos permitirá tomar una idea del tipo de usuario y del escenario de uso.

2.1.1 Ficha de persona

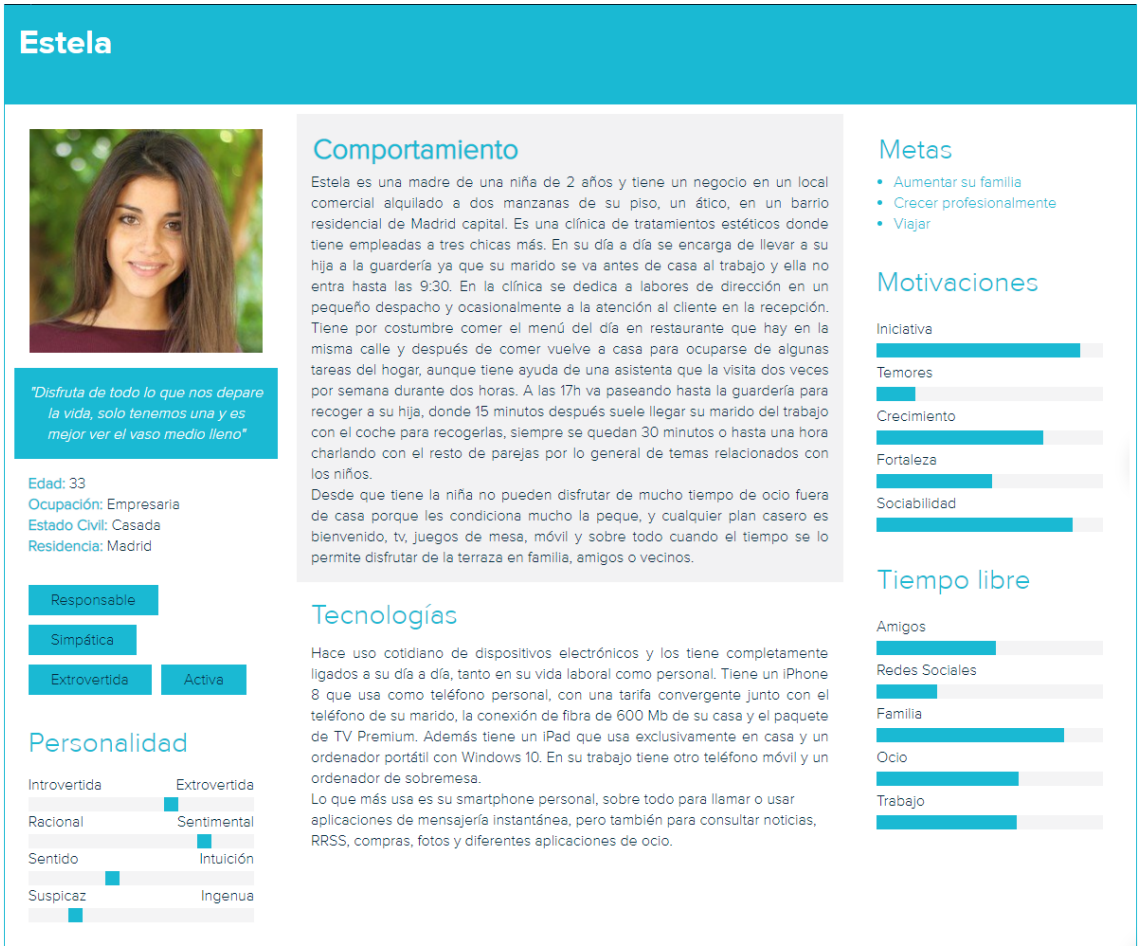


Figura 5. Ficha Persona. Estela

Para poder obtener información que nos pueda servir para tomar decisiones de diseño de nuestra aplicación comenzaremos realizando una entrevista a Estela, como una posible usuaria de la aplicación. Para ello hemos decidido realizar una entrevista cerrada ya que la funcionalidad core está clara y únicamente queremos centrarnos en aspectos concretos de usabilidad.

Podemos ver la transcripción completa de la entrevista en el *Anexo 1* de este documento.

2.2. Diseño conceptual

En este apartado describiremos varios escenarios en los que se produciría interacción con nuestra aplicación.

2.2.1. Escenarios de casos de uso

Escenario 1, Estela:

Estela está comiendo en el restaurante de todos los días consultado el móvil entre plato y plato. Buscando entre las aplicaciones instaladas encuentra la aplicación que descargó para estar enterada de las próximas lluvias de estrellas que pudiera ver desde Madrid. En su momento descargó la aplicación porque una noche mientras transcurría una cena con amigos en su terraza comenzaron a caer estrellas y todos pasaron un buen rato tomando algo y tumbados en las hamacas que allí tiene. Esa noche decidieron quedar siempre que pudieran para ver desde ahí las lluvias de estrellas.
Pues en ese momento se percató de que no tenía activadas las notificaciones y esa misma noche había una lluvia de estrellas, con lo que se decepcionó porque no le daría tiempo a organizar nada ni a avisar a sus amigos. En ese momento dejó activadas las notificaciones permanentemente con una semana de antelación.

Escenario 2, Mario:

Mario ha llegado a casa tras un día de trabajo y está charlando con su pareja mientras preparan la cena. En ese momento suena una notificación en su móvil, se trata de una aplicación que alerta de próximos eventos astronómicos, en el mensaje de la alerta indica: “Próximo eclipse lunar: sábado 25 de junio, 21:45h”. Mario lee la nota en voz alta y comienza a hablar sobre el tema con su pareja y a los dos le parece buena idea hacer una escapada ese fin de semana a una casa rural y disfrutar de ese memento tranquilamente.
Mientras tanto, Mario entre en la aplicación para conocer los detalles del evento, como la hora, la visibilidad, etc.

Escenario 3, Susana:

Un lunes más Susana se encuentra en la facultad de educación son las 11 de la mañana y se dirige con una compañera a el aula de la segunda asignatura de la mañana. Durante el paseo por el pasillo van hablando de su fin de semana, María, la amiga, comenta a Susana que la noche del sábado salió con su pareja al campo para ver la lluvia de estrellas que habían anunciado ese mismo día en las noticias, a lo que Susana respondió con curiosidad ya que le hubiera gustado verlo también pero no se enteró. A continuación, María le muestra una aplicación en su móvil que permite programar notificaciones de estos y otros eventos astronómicos. Susana se apresura a descargar la *app* antes de entrar en clase para que no se le olvide.
Esa misma tarde, Susana accede a la aplicación para curiosear. Rápidamente se hace con el control gracias a la sencillez en la presentación de la información. Sintió curiosidad por saber que eventos se producirían las fechas

próximas a su cumpleaños por lo que filtro el contenido por fecha y así pudo salir de dudas.

Tras los primeros pasos de análisis podemos empezar a esbozar las primeras decisiones de diseño:

- Presentar información básica del contenido.
- Funcionalidad estrella → notificaciones.
- Posibilidad de guardar información/fotos relacionadas a un evento.
- Posibilidad de ampliar en versiones posteriores sistemas de share.
- Filtrado de eventos por rango de fechas.

2.2.2 Identificación de pantallas

A continuación, describiremos las ventanas identificadas y el inventario de contenidos que consideramos necesario según lo expuesto por el usuario y la funcionalidad necesaria para el objetivo de la aplicación.

Hemos identificado cada pantalla con un código identificativo que usaremos en adelante para referenciarlas. En la siguiente tabla relacionamos el código de ventana con los requisitos que impactan en su diseño.

Ventana	Requisito relacionado
P.AT.00.00. Ventana Inicial	RF.TFM.001 RF.TFM.002 RF.TFM.003 RNF.TFM.001 RNF.TFM.002
P.AT.01.00. Ventana de listado de Eclipses	RF.TFM.001 RF.TFM.002 RF.TFM.003 RF.TFM.004 RF.TFM.005 RF.TFM.007 RF.TFM.008 RNF.TFM.001 RNF.TFM.002
P.AT.01.01. Ventana de detalle de Eclipse	RF.TFM.001 RF.TFM.002 RF.TFM.009 RF.TFM.010 RNF.TFM.001 RNF.TFM.002
P.AT.01.02. Ventana de mapa de visualización de Eclipse	RF.TFM.001 RF.TFM.002 RF.TFM.010 RNF.TFM.001 RNF.TFM.002
P.AT.02.00. Ventana de listado de Fases Lunares	RF.TFM.001 RF.TFM.002 RF.TFM.003

	RF.TFM.004 RF.TFM.006 RF.TFM.007 RF.TFM.008 RNF.TFM.001 RNF.TFM.002
P.AT.02.01. Ventana de detalle de Fase Lunar	RF.TFM.001 RF.TFM.002 RF.TFM.009 RNF.TFM.001 RNF.TFM.002
P.AT.03.00. Ventana de listado de Lluvias de Estrellas	RF.TFM.001 RF.TFM.002 RF.TFM.003 RF.TFM.004 RF.TFM.005 RF.TFM.007 RF.TFM.008 RNF.TFM.001 RNF.TFM.002
P.AT.03.01. Ventana de detalle de Lluvia de Estrellas	RF.TFM.001 RF.TFM.002 RF.TFM.009 RNF.TFM.001 RNF.TFM.002
P.AT.04.01. Ventana de Ayuda	RF.TFM.001 RF.TFM.002 RF.TFM.011 RNF.TFM.001 RNF.TFM.002
P.AT.05.01. Ventana de Ajustes	RF.TFM.001 RF.TFM.002 RF.TFM.011 RNF.TFM.001 RNF.TFM.002
P.AT.06.01. Ventana de Tipos de Alertas	RF.TFM.001 RF.TFM.002 RF.TFM.011 RNF.TFM.001 RNF.TFM.002
P.AT.06.02. Ventana de Alertas de Eclipses	RF.TFM.001 RF.TFM.002 RF.TFM.012 RNF.TFM.001 RNF.TFM.002
P.AT.06.03. Ventana de Alertas de Fases Lunares	RF.TFM.001 RF.TFM.002 RF.TFM.012 RNF.TFM.001 RNF.TFM.002
P.AT.06.04. Ventana de Alertas de Lluvia de Estrellas	RF.TFM.001 RF.TFM.002 RF.TFM.012 RNF.TFM.001

	RNF.TFM.002
P.AT.06.05. Ventana de configuraciones generales	RF.TFM.001 RF.TFM.002 RNF.TFM.001 RNF.TFM.002

Tabla 7. Relación de ventanas y requisitos

2.2.3 Definición de pantallas

P.AT.00.00. Ventana inicial:
Pantalla estructurada con un *TabBar* de tres pestañas que permita almacenar el contenido principal de cada tipo de evento. La elección de este tipo de menú se debe a lo intuitivo del mismo, y a que permite visualizar en todo momento cada una de las pestañas y moverse entre ellas bien pinchando sobre una o deslizando lateralmente la ventana.

Este *TabBar* contendrá dos elementos comunes, un botón de Menú y un botón Home para volver a la ventana principal que serán comunes para todas las ventanas de la aplicación.

Desglosamos el contenido de cada pestaña.

P.AT.01.00. Ventana de listado de Eclipses:
Desplegable: Para seleccionar un año determinado como filtro de presentación. Por defecto no estará informado y se mostrará la información listada por años.
Listado de elementos: cada elemento estará compuesto de una imagen representativa del tipo de eclipse y de un cuadro de texto con 2 líneas, una para la descripción y otra para la fecha. Al pulsar un elemento de la lista se accederá a su P.AT.01.01. Ventana de detalle de Eclipse.

P.AT.01.01. Ventana de detalle de Eclipse:
Imagen: Incluir imagen representativa del tipo de eclipse
Label Titulo: Texto con el título descriptivo del tipo de eclipse.
Label Fecha: Texto con la fecha del eclipse.
Temporizador: Campo para marcar una cuenta atrás hasta la fecha del eclipse.
Botón Mapa: Botón para acceder a la P.AT.01.02. Ventana de mapa de visualización de Eclipse.
Check + Texto: Para poder activar de forma rápida una notificación sobre el evento.

P.AT.01.02. Ventana de mapa de visualización de Eclipse:
Mapa: Mostrar sobre el mapa la zona de influencia del eclipse.

P.AT.02.00. Ventana de listado de Fases Lunares:
Desplegable: Para seleccionar mes y año determinado como filtro de presentación. Por defecto estará informado el mes en curso.
Tabla de botones: Necesaria para representar el calendario, será una cuadricula con botones en las celtas que contendrán la información lunar de cada día. Pulsando en uno de ellos se accederá a la ventana detalle.

P.AT.02.01. Ventana de detalle de Fase Lunar:
Imagen: Incluir imagen representativa del tipo de Fase Lunar.
Label Titulo: Texto con el título descriptivo del tipo de Fase Lunar.
Label Fecha: Texto con la fecha de la Fase Lunar.
Temporizador: Campo para marcar una cuenta atrás hasta la fecha de la Fase Lunar.
Check + Texto: Para poder activar de forma rápida una notificación sobre el evento.

P.AT.03.00. Ventana de listado de Lluvias de Estrellas:
Desplegable: Para seleccionar mes y año determinado como filtro de presentación. Por defecto no estará informado y se mostrará la información listada por meses.
Listado de elementos: Cada elemento estará formado por tres líneas de texto, la principal de mayor tamaño con el nombre del fenómeno, la segunda con la fecha, y una tercera con símbolos de estrellas, de 1 a 5 dependiendo de la visibilidad estimada del evento. Pulsando sobre un elemento de la lista se accederá a la ventana de detalle.

P.AT.03.01. Ventana de detalle de Lluvia de Estrellas:
Label Titulo: Texto con el título descriptivo del evento.
Label Fecha: Texto con el rango de fechas en el que se podrá ver el evento.
Gráfico: Mostrará gráficamente la visibilidad del fenómeno en el periodo de tiempo que dure.
Check + Texto: Para poder activar de forma rápida una notificación sobre el evento.

MENU:
Será un desplegable que contendrá los accesos a la ayuda y configuraciones.

P.AT.04.01. Ventana de Ayuda:
Contendrá elementos de tipo texto únicamente. La información ser lo más concisa posible y estará estructurada por bloques dando respuesta a cada funcionalidad permitida en la *app*.

P.AT.06.01. Ventana de Tipos de Alertas:
Esta configuración permite activar y desactivar alertas por las tipologías de fenómeno que están recogidas dentro de cada tipo de evento.

P.AT.06.02. Ventana de Alertas de Eclipses:
Texto descriptivo + botón de activación/desactivación: Habrá un listado de opciones con los diferentes tipos de fenómeno que permitirá activar y desactivar

P.AT.06.03. Ventana de Alertas de Fases Lunares:
Texto descriptivo + botón de activación/desactivación: Habrá un listado de opciones con los diferentes tipos de fenómeno que permitirá activar y desactivar

P.AT.06.04. Ventana de Alertas de Lluvia de Estrellas:
Texto descriptivo + botón de activación/desactivación: Habrá un listado de opciones con los diferentes tipos de fenómeno que permitirá activar y desactivar

P.AT.06.05. Ventana de configuraciones generales:
Texto descriptivo + desplegable: Para configurar con que antelación queremos que se envíe cada tipo de alerta. El replegable permitirá selección por unidades de tiempo. (horas, días…)

P.AT.05.01. Ventana de Ajustes:
Texto descriptivo + desplegable: Para permitir modificar los ajustes de que disponemos, como ciudad o idioma
Texto descriptivo + botón de activación/desactivación: para activar o desactivar las notificaciones.

2.3. Prototipado

Con la idea bastante clara, en este punto ya podemos esbozar un primer borrador en papel del flujo de ventanas que queremos implementar. Sobre este diagrama se han ido añadiendo cambios y modificaciones necesarias hasta dejar una solución que sobre el papel consideramos robusta.

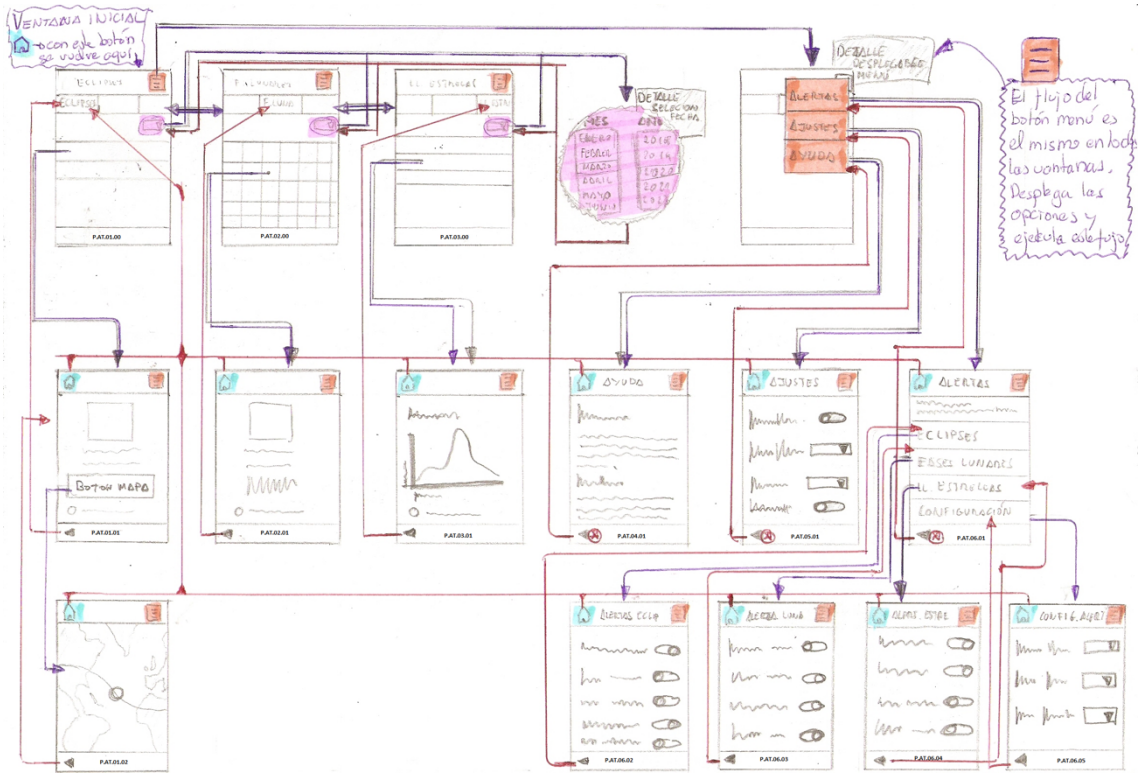


Figura 6. Borrador de Diagrama de ventanas de aplicación

Con el diagrama de flujos claro podemos comenzar a definir a más bajo nivel los detalles del diseño de la aplicación.

De entre los diferentes modelos de representación, la que más adapta a nuestro diseño según las preferencias del usuario es la jerárquica ya que tiene varios niveles bien definidos y no llega a superar los 3 niveles en ningún momento.

En los elementos seleccionados para conformar cada ventana se ha tenido en cuanta la sencillez de uso y el factor intuitivo en cada uno de ellos. Ya que si muestras una lista con formato de botón intuitivamente entenderás que si pinchas entraras al detalle, o que si ponemos un botón de activación detrás de un texto estaremos ejecutando la acción que indique el texto.

Tampoco es casual la elección de los iconos, se ha pensado en los más representativos y habituales para cada tipo de acción que se va a ejecutar al pulsarlos, como la casita que te devuelve a la Home o el radio botón de habilitar que permite determinadas notificaciones.

Los textos serán escasos y muy concisos, únicamente mostrando la información indispensable. Se tendrá en cuanta una jerarquía de texto donde dar relevancia con negrita y tamaños más grandes a algunos elementos.

Por último, a tener en cuenta la forma de mostrar las notificaciones al usuario. Pretendemos que sean lo menos intrusivas. Se mostrarán con el tiempo de antelación que ha seleccionado el usuario, en la parte superior de la pantalla si el usuario está usando el teléfono o en la pantalla bloqueada en caso contrario. Estas notificaciones contendrán el icono de la aplicación y un mensaje indicando el tipo de evento y la fecha de este.

En este punto es el momento de realizar un prototipo de baja fidelidad con una herramienta digital. La intención es dar un aspecto más realista a los bocetos realizados a mano que nos permita hacernos una idea más aproximada del resultado final y tomar decisiones de diseño anticipándonos a la fase de implementación.

Estas son las ventanas principales de nuestra aplicación realizadas con la herramienta *Balsamiq*[3]

Ventanas de menú principal con pestañas de cada evento:

[3] Herramienta para creación de Mocks https://balsamiq.com/wireframes/

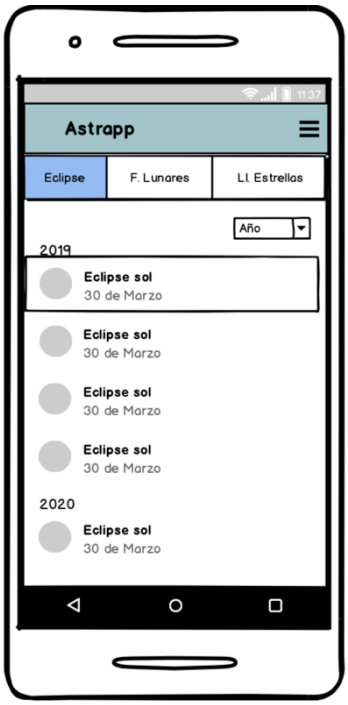


Figura 7. Mock de P.AT.01.00. Ventana de listado de Eclipses

Figura 8. Mock de P.AT.02.00. Ventana de listado de Fases Lunares

Figura 9. Mock de P.AT.03.00. Ventana de listado de Lluvias de Estrellas

Ventanas de cada tipo de detalle de en evento seleccionado:

Figura 10. Mock de P.AT.01.01. Ventana de detalle de Eclipse

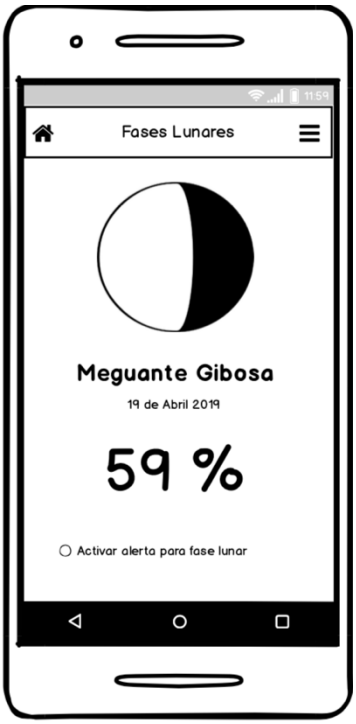


Figura 11. Mock de P.AT.02.01. Ventana de detalle de Fase Lunar

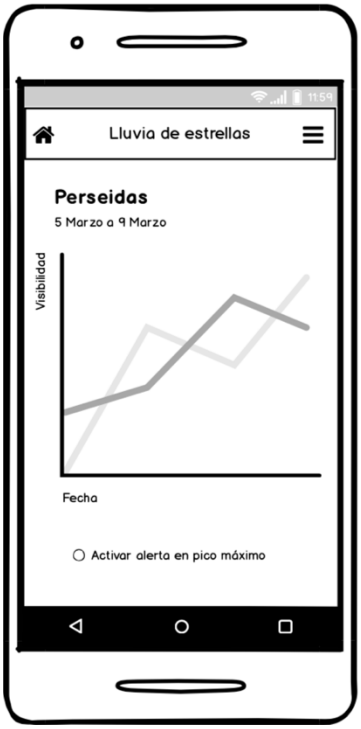


Figura 12. Mock de P.AT.03.01. Ventana de detalle de Lluvia de Estrellas

Por último, concluimos la fase de prototipado realizando un prototipo digital interactivo de alta definición que simule tanto el aspecto como la funcionalidad de la aplicación del modo más aproximado posible. Para ello hemos usado la herramienta *JustInMind* [4]. El prototipo se encuentra como adjunto en los ficheros entregados junto con la memoria en la ruta *prototipo/Prototipo.vp* y se puede ejecutar con la propia herramienta *JustInMind* pulsando en Simulate.

[4] Herramienta online para la creación de prototipos de aplicaciones https://www.justinmind.com/

En el *Anexo 2* de este documento podemos ver una captura de cada una de las pantallas creadas en el prototipo.

2.4. Evaluación

2.4.1. Análisis Heurístico

El proceso de prototipado necesita de criterios de evaluación que dar validez a las decisiones de diseño y funcionalidad tomadas en esta fase. En primer lugar, usaremos el método de análisis heurístico que implica la revisión del diseño teniendo en cuenta diferentes reglas y principios de usabilidad previamente establecidos. Nuestro análisis tomará como referencia las diez reglas heurísticas de Jakob Nielsen[5] para el diseño de interfaces que, aunque en un principio fueron pensadas para el diseño web, fácilmente pueden ser adaptadas al diseño de cualquier interfaz, en nuestro caso móvil.

• Visibilidad del estado del sistema: *el sistema debe informar en todo momento al usuario de lo que está ocurriendo, dando respuestas en un tiempo razonable.*

Este es un aspecto en el que ha hecho hincapié desde un principio y que nuestro prototipo ha cumplido. El menú de pestañas marca de otro color la pestaña que está seleccionada, y en cada subventana se añade el título descriptivo al *ToolBar* de la aplicación.

• Relación entre el sistema y el mundo real: *el sistema debe hablar el lenguaje de los usuarios y seguir las convenciones del mundo real, y no hablar el lenguaje de las máquinas.*

Tras una primera iteración se ha cambiado el icono de vuelta a la ventana principal por un botón Home que es más intuitivo. También se decide añadir una imagen representativa de cada fenómeno en el listado de estos.

• Control y libertad del usuario: *hay ocasiones en que los usuarios elegirán las funciones del sistema por error y necesitarán una «salida de emergencia» claramente marcada. El sistema debe permitir las opciones de deshacer y rehacer.*

Todas las ventanas accedidas tienen la opción de volver atrás desde el *TabBar* o bien desde el botón Home.
En cuanto a los cambios en la programación de alertas se ha decido implementar botones de check o activaciones fácilmente revocables.
Tras esta revisión se decide introducir la opción de restaurar valores iniciales.

[5] Enlace a fuente de las reglas heurísticas de Jakob Nielsen. http://www.braintive.com/10-reglas-heuristicas-de-usabilidad-de-jakob-nielsen/

• Consistencia y estándares: *los usuarios no deberían cuestionarse si acciones, situaciones o palabras diferentes significan en realidad la misma cosa. La interfaz ha de seguir las convenciones de la plataforma o sistema operativo.*

El análisis de esta regla nos ha ayudado a decidir la forma de mostrar las ventanas de ajustes, replicando el aspecto y modo de uso de las existentes en el propio sistema Android.

• Prevención de errores: *un buen diseño que evita los errores será siempre preferible a un buen mensaje de error.*

Está regla a sacado una validación a tener en cuenta en tiempo de desarrollo, no deberemos dejar seleccionar una fecha anterior a la actual para programar una alerta.

• Reconocimiento antes que recuerdo: *las instrucciones para el uso del sistema deben estar a la vista o ser fácilmente accesibles cuando sea necesario.*

Tras esta iteración entendemos que debemos añadir un indicador visual en los listados de fenómenos que indiquen se tiene ya una alerta programada.

• Flexibilidad y eficiencia de uso: *el sistema debe brindar caminos con instrucciones claras para los usuarios novatos, sin dificultar el camino a los usuarios avanzados. Permite atajos al usuario experto.*

El botón Home se añade para cumplir el deseo del usuario que necesite salir de varios niveles de jerarquía hasta el inicio de una sola vez.

• Estética y diseño minimalista: *los diálogos no deben contener información irrelevante. En un móvil, por ejemplo, cada pedazo de información extra compite con lo importante y resta espacio en pantalla.*

La aplicación cumple óptimamente esta regla, solo mostramos títulos he información concreta y relevante. Las ventanas nos están recargadas y en un vistazo se comprende el contenido.

• Ayudar a los usuarios a reconocer, diagnosticar y recuperarse de errores: *los mensajes de error se deben entregar en un lenguaje claro y simple, indicando de forma precisa el problema y mostrar una solución.*

Los mensajes de error o informativos se redactan de forma clara y se dará la opción de corregir el error.

• Ayuda y documentación: *la ayuda debe ser fácil de encontrar, estar dirigida a las tareas de los usuarios y ser entendible y breve. Mucho mejor si está contextualizada: que se relacione con el paso en el que se encuentra el usuario.*

El menú de ayuda está accesible en todo momento desde el *ToolBar*.

2.4.2. Test con usuarios

Necesitamos tener *feedback* de usuarios testando la aplicación sobre el prototipo interactivo. Esto nos ayudará a tomar decisiones en función de las dudas que presente o comentarios que pueda hacer.
Debemos animar al usuario a que comparta en voz alta sus impresiones, pero sin llegar a interferir en sus decisiones.

Hemos organizado el test en dos fases.
En un primer momento dejaremos que el usuario descubra libremente la aplicación, durante unos 3 minutos, donde recogeremos cualquier comportamiento que consideremos interesante para la mejora de aplicación.
La segunda fase consistirá en proponer al usuario una serie de acciones que tiene que conseguir reproducir sobre la aplicación, estas son:

- Activar alerta para una lluvia de estrellas en concreto.
- Filtrar las fases lunares de un mes en concreto.
- Activar alertas para todos los fenómenos del tipo eclipse solar total.
- Cambiar idioma de la aplicación.
- Consultar el mapa de visualización de un eclipse.
- Consultar la ayuda.

Tras realizar el test a varios usuarios las observaciones que hemos recogido son las siguientes:

- En la primera parte del test vemos que los usuarios entienden rápidamente la funcionalidad de los menús y las navegaciones que permiten. Atendiendo a algunos comentarios recogemos los siguientes puntos de mejora:
    - El literal “Fecha” no es intuitivo a la hora de filtra una lista, debernos explicar mejor la finalidad de esta funcionalidad. Cambiar por “Buscar por fecha”
    - Añadir una pequeña descripción en los menús de alertas que indique el resultado de las acciones que ahí pueden gestionar.

- Sobre la segunda fase de test guiado estas son las conclusiones:
    - Activar alerta para una lluvia de estrellas en concreto: todos los usuarios concluyeron la prueba con éxito en un tiempo razonable.
    - Filtrar las fases lunares de un mes en concreto: la mayoría de los usuarios ejecutaron la acción usando el filtrado que ofrece la ventana. Un usuario llegó a la solución deslizando mes a mes el contenido hasta llegar al solicitado. Cuando le preguntamos porque no había usado el filtro indicó que no lo había entendido con ese fin, con lo que reforzamos la decisión de cambio de litera.
    - Activar alertas para todos los fenómenos del tipo eclipse solar total: hubo un comportamiento similar en varios los usuarios, su primera acción fue buscar un evento de este tipo y analizar donde realizar la acción. Al no encontrarla navegaron hasta el

menú correcto de Alertas y encontraron la opción sin mayor problema. Este comportamiento nos hace plantear la posibilidad de añadir a nivel de detalle de un fenómeno la posibilidad de activar las alertas para todos los fenómenos de ese mismo tipo.

- Cambiar idioma de la aplicación: Todos los usuarios buscaron el menú de ajustes y llegaron de inmediato a la solución.
- Consultar el mapa de visualización de un eclipse: Llegaron a la solución de una forma natural, sin errores de navegación y a la primera.
- Consultar la ayuda: Todos los usuarios optaron intuitivamente por desplegar el menú superior donde efectivamente se encuentra la ayuda.

2.5. Casos de Uso

2.5.1. Diagramas UML de Casos de Uso

DU.001. Diagrama de Sistema

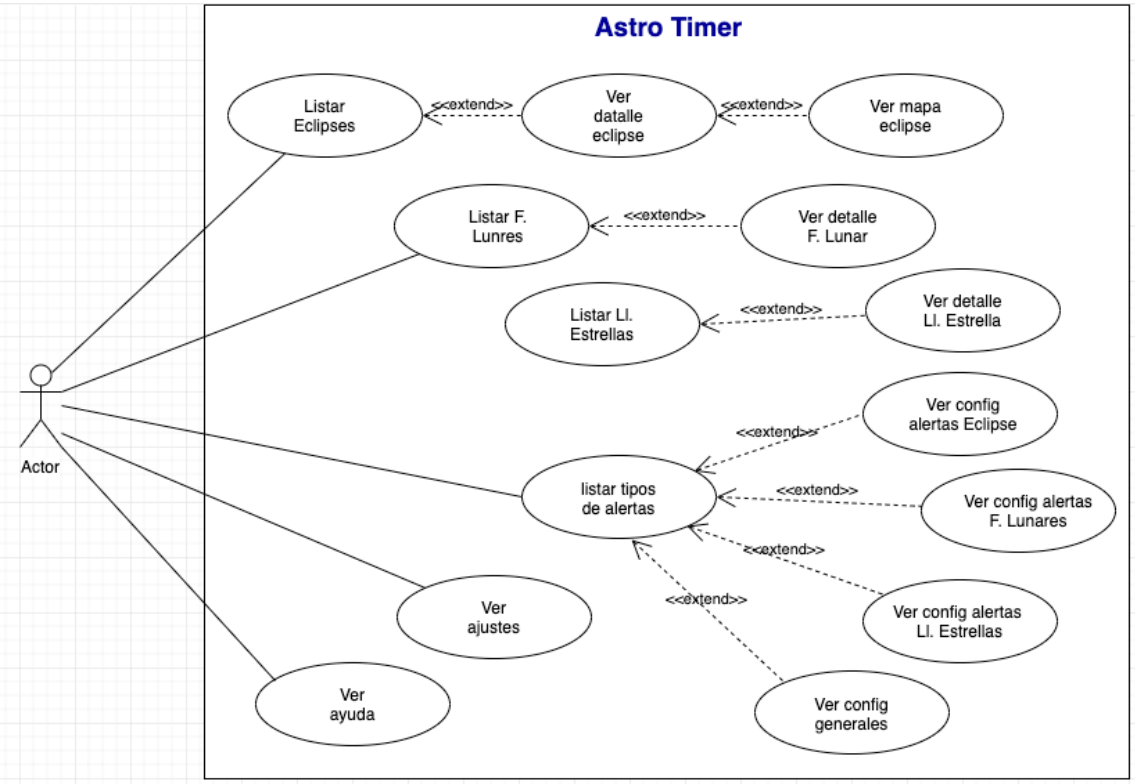


Figura 13. DU.001. Diagrama de Sistema

DU.002. Diagrama de Listado de Eclipses

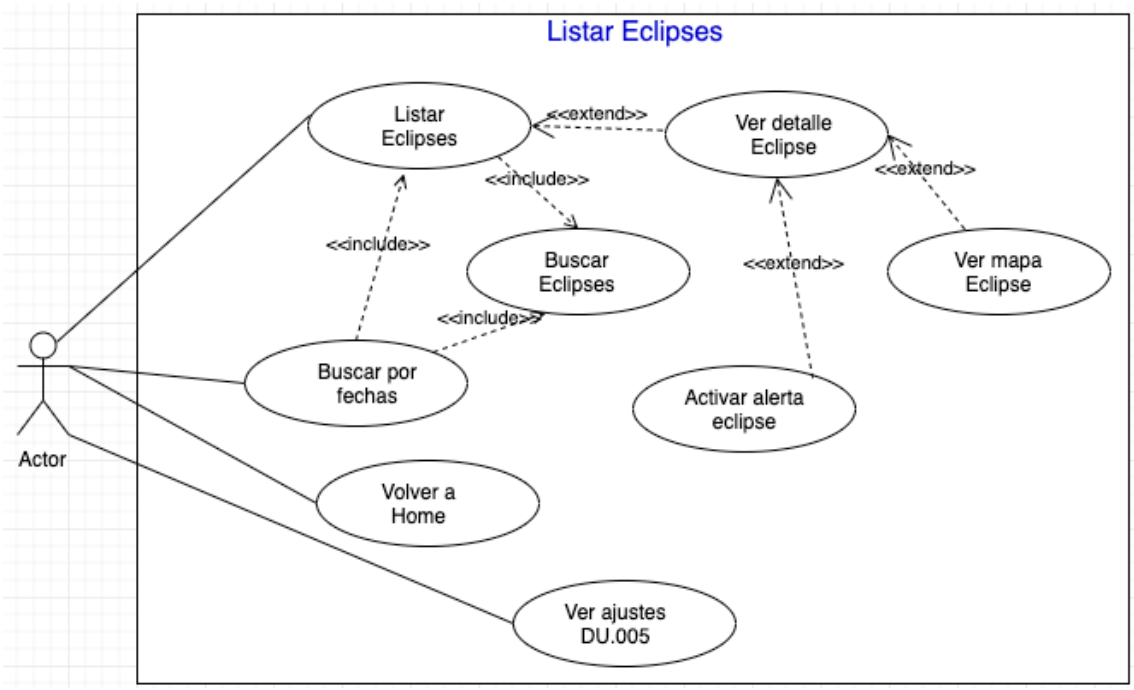


Figura 14. DU.002. Diagrama de Listado de Eclipses

DU.003. Diagrama de Listado de Fases Lunares

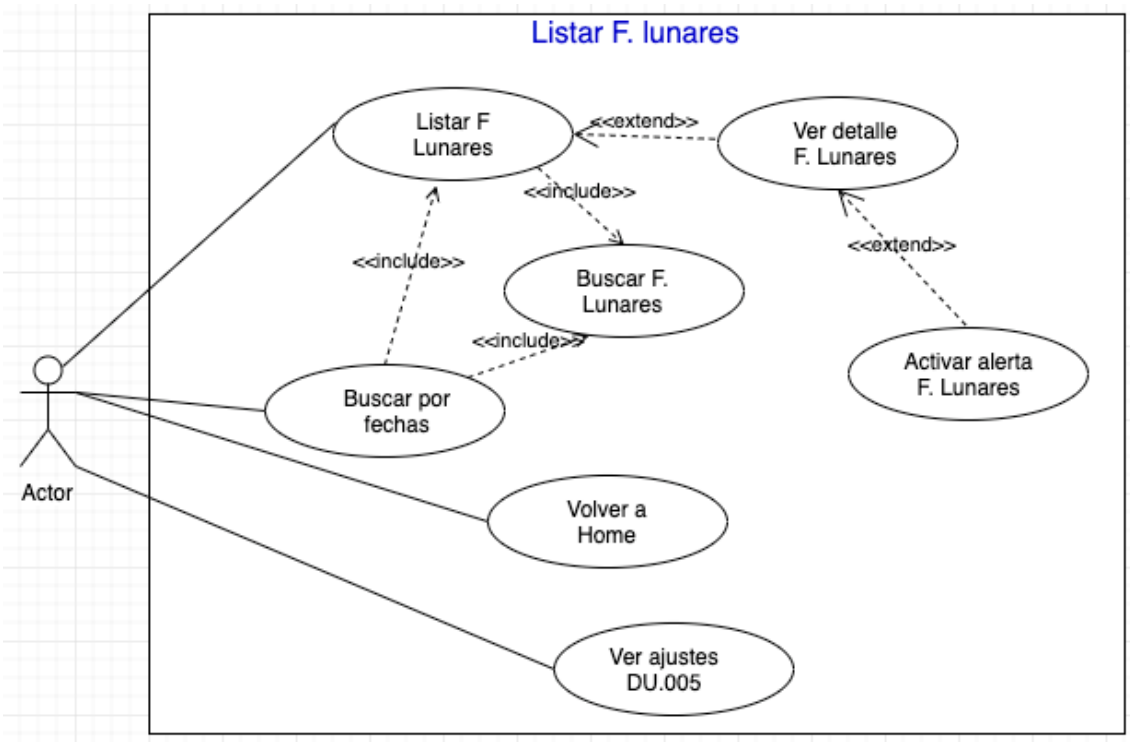


Figura 15. DU.003. Diagrama de Listado de Fases Lunares

DU.004. Diagrama de Listado de Lluvia de estrellas

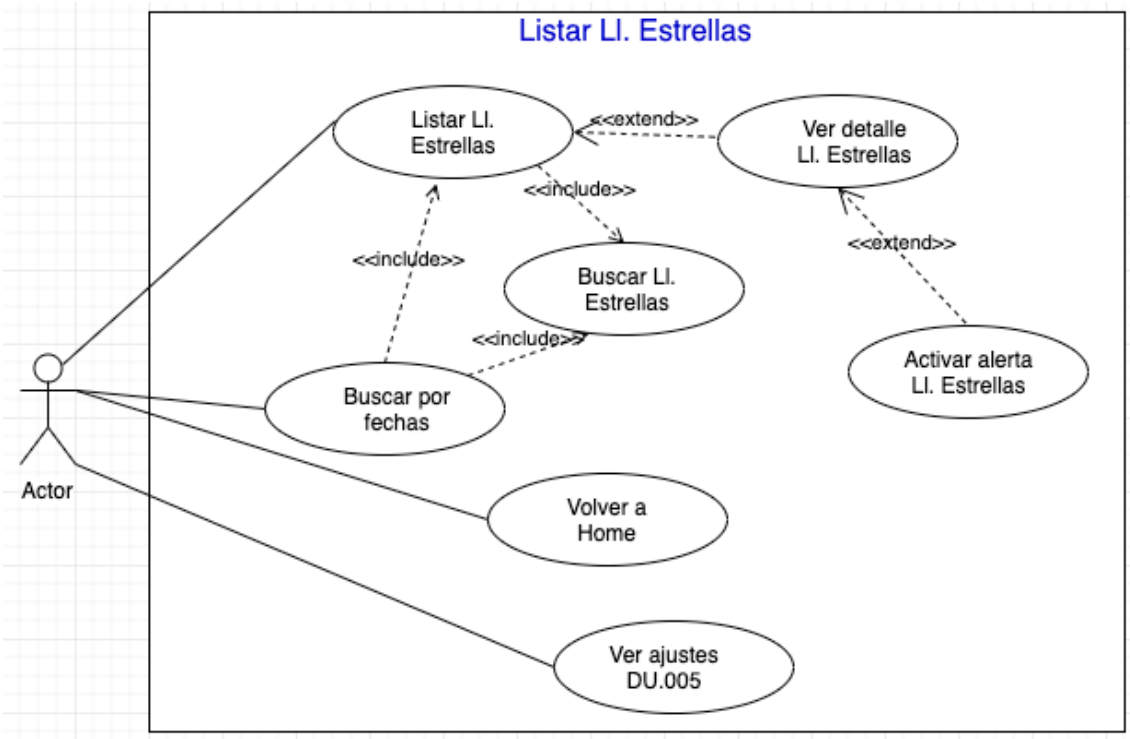


Figura 16. DU.004. Diagrama de Listado de Lluvia de estrellas

DU.005. Diagrama de Menú

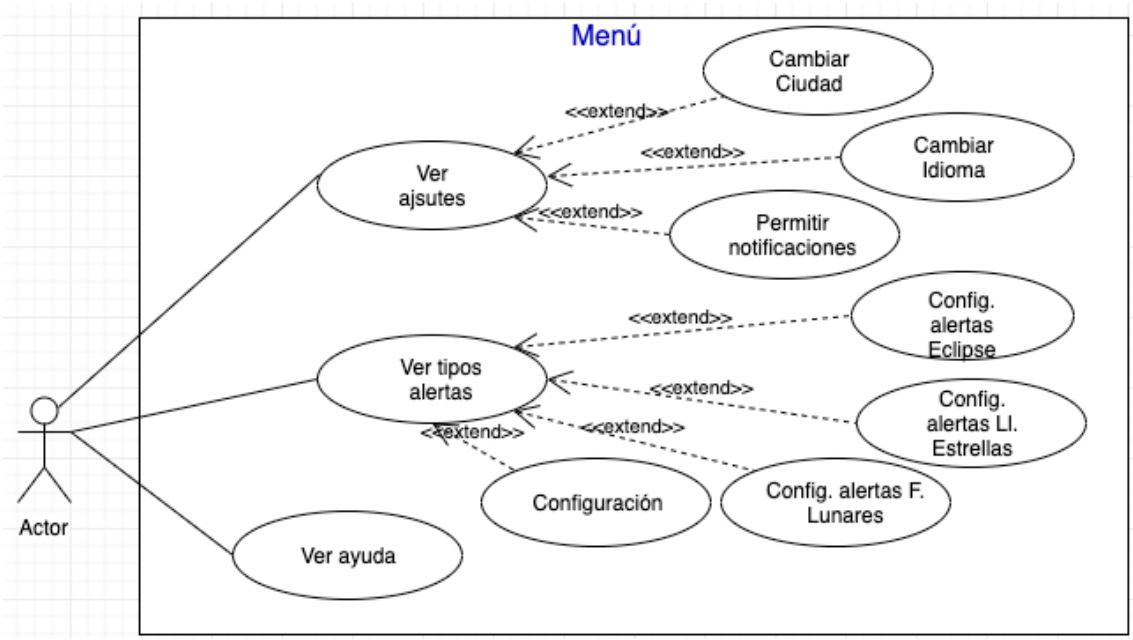


Figura 17. DU.005. Diagrama de menú

2.5.2. Listado de Casos de Uso

CU.001 Listar Eventos por tipología

Caso de Uso:	Listar eventos por tipología
Código:	*CU.001*
Actores:	*Usuario*
Descripción:	*El usuario quiere listar los eventos de un determinado tipo. Este caso de uso sirve para las diferentes tipologías de evento que se muestran en la aplicación.*
Diagrama relacionado:	*DU.002, DU.003 y DU.004*
Precondiciones:	El usuario ha accedido a la aplicación
Postcondiciones:	El usuario visualiza el listado de eventos en la pantalla

Escenario principal	
Usuario	**Sistema**
1. Pulsa sobre una pestaña del Tab Bar	2. Conecta con el servidor y hace la petición de datos 3. El servidor devuelve la información 4. Se muestra la información en pantalla

Tabla 8. CU.001. Listar eventos por tipología

CU.002 Acceso a mapa de eclipse

Caso de Uso:	Acceso a mapa de eclipse
Código:	*CU.002*
Actores:	*Usuario*
Descripción:	*El usuario quiere consultar el área de visibilidad de un eclipse*
Diagrama relacionado:	*CU.002*
Precondiciones:	El usuario ha accedido a la aplicación
Postcondiciones:	El sistema muestra el mapa de visibilidad del eclipse seleccionado

Escenario principal	
Usuario	**Sistema**
1. Pulsa sobre la pestaña de eclipses	2. Conecta con el servidor y hace la petición

	de datos 3. El servidor devuelve la información
5. Pulsa sobre un eclipse de la lista	4. Se muestra la información en pantalla
7. Pulsa sobre el botón de Mostrar área de visibilidad	6. Carga la información del eclipse y la muestra en ventana
	8. Muestra el mapa del eclipse

Tabla 9. CU.002. Acceso a mapa de eclipse

CU.003 Filtrado de eventos por fecha

Caso de Uso:	Filtrado de tipos de evento por fecha
Código:	*CU.003*
Actores:	*Usuario*
Descripción:	*El usuario quiere filtrar por un determinado rango de fecha y listar los eventos de un determinado tipo*
Diagrama relacionado:	*DU.002, DU.003 y DU.004*
Precondiciones:	El usuario está en la pestaña de listado de eventos de un tipo
Postcondiciones:	El sistema solo muestra los eventos de ese tipo cuya fecha esta comprendida entre el rango seleccionado
Escenario principal	
Usuario	**Sistema**
1. Pulsa sobre el buscador de fechas	2. Muestra el calendario donde seleccionar fecha desde - hasta
3. Selecciona las fechas 4. Pulsa buscar	5. Se muestran los resultados filtrados por fecha

Tabla 10. CU.003. Filtrado de eventos por fecha

CU.004 Ver detalle de eventos

Caso de Uso:	Ver detalle de eventos
Código:	*CU.004*
Actores:	*Usuario*
Descripción:	*El usuario quiere ver el detalle de un evento de un determinado tipo*
Diagrama relacionado:	*DU.002, DU.003 y DU.004*

Precondiciones:	El usuario ha accedido a la aplicación
Postcondiciones:	El sistema muestra el detalle del evento seleccionado.
Escenario principal	
Usuario	**Sistema**
1. Pulsa sobre la pestaña del Tab Bar del tipo de evento que quiere consultar	2. Conecta con el servidor y hace la petición de datos 3. El servidor devuelve la información 4. Se muestra la información en pantalla
5. Pulsa sobre un evento de la lista	6. Carga la información del evento y la muestra en ventana

Tabla 11. CU.004. Ver detalle de eventos

CU.005 Cambiar idioma de la aplicación

Caso de Uso:	Filtrado de tipos de evento por fecha
Código:	*CU.005*
Actores:	*Usuario*
Descripción:	*El usuario quiere listar los eventos de un determinado tipo*
Diagrama relacionado:	*DU.005*
Precondiciones:	El usuario ha accedido a la aplicación
Postcondiciones:	La aplicación cambia el idioma
Escenario principal	
Usuario	**Sistema**
1. Pulsa sobre el menú desplegable	2. Muestra las opciones del menú
3. Pulsa sobre la opción Ajuste	3. Muestra las opciones de ajustes
5. Pulsa sobre el desplegable de Idioma	6. Muestra el listado de idiomas disponibles
7. Selecciona un idioma de la lista	8. Guarda la opción seleccionada y cambia el idioma de los textos

Tabla 12. CU.005. Cambiar idioma de la aplicación

CU.006 Activar alerta de un determinado evento

Caso de Uso:	Activar alerta de un determinado tipo
Código:	*CU.006*
Actores:	*Usuario*
Descripción:	*El usuario quiere activar la alerta de un evento en concreto*
Diagrama relacionado:	*DU.005*
Precondiciones:	El usuario ha accedido a la ventana de detalle de un evento que no tiene la alerta activada
Postcondiciones:	Se activa la alerta para el evento seleccionado

Escenario principal	
Usuario	**Sistema**
1. Pulsa sobre el check de activación de alerta.	2. Guarda y programa la alerta

Tabla 13. CU.006. Activar alerta de un determinado evento

CU.007 Desactivar todas las notificaciones

Caso de Uso:	Desactivar todas las notificaciones
Código:	*CU.007*
Actores:	*Usuario*
Descripción:	*El usuario quiere desactivar todas las notificaciones*
Diagrama relacionado:	*DU.005*
Precondiciones:	El usuario ha accedido a la aplicación y ha entrado al menú de ajustes
Postcondiciones:	El sistema desactiva los avisos programadas, pero no elimina las configuraciones

Escenario principal	
Usuario	**Sistema**
1. Pulsa el botón de desactivación de notificaciones	2. El sistema desactiva las alertas

Tabla 14. CU.007. Desactivar todas las notificaciones

2.6. Diseño de la arquitectura

2.6.1. Diagrama UML de Base de Datos

El sistema de almacenado de información estará repartido en 3 capas.
La capa de Servidor, donde se encontrará la información actualizada de todos fenómenos. Esta información estará almacenada en un fichero JSON.
La capa de Base de Datos, donde guardaremos la relación de alertas configuradas por el usuario, así como aspectos propios de la configuración de la aplicación y la relación de imágenes con sus respectivos fenómenos.
Por último, la capa de ficheros, donde guardaremos las propias imágenes necesarias para mostrar como representación de un fenómeno.

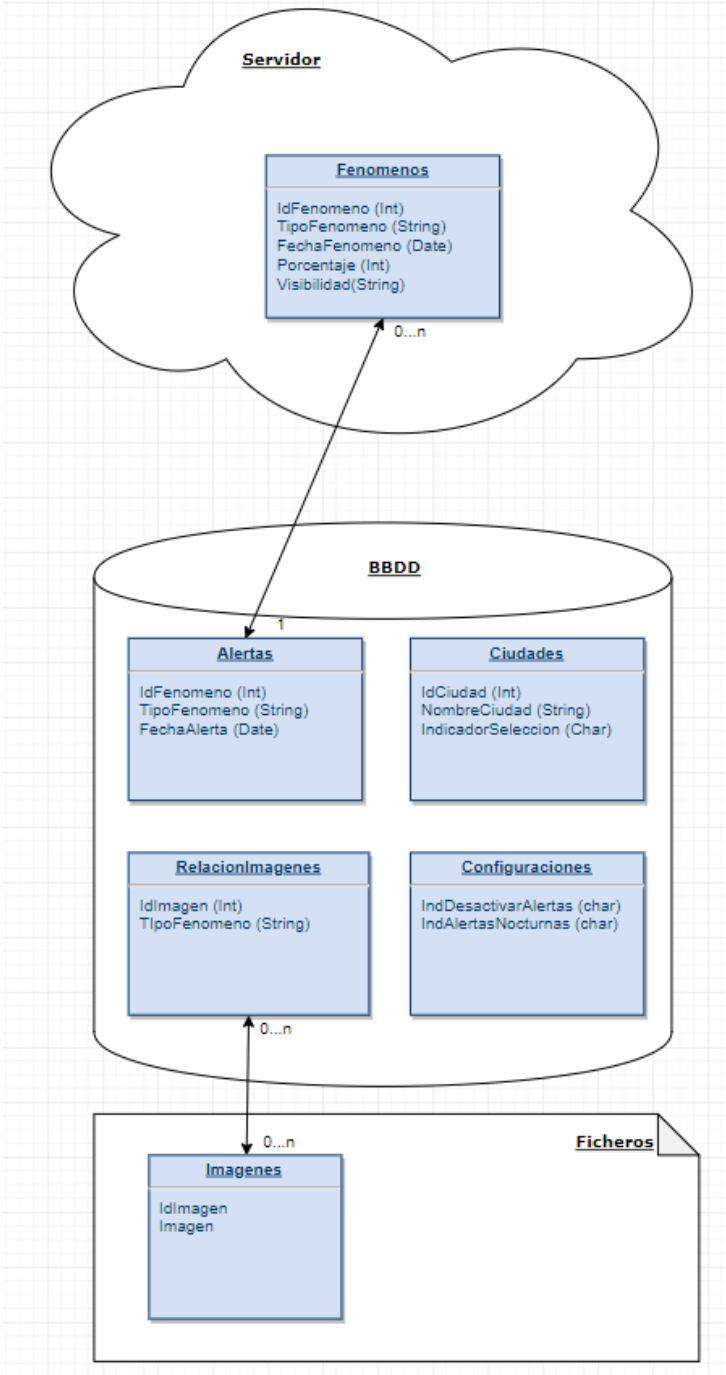


Figura 18. Diagrama de Base de Datos

2.6.2. Diagrama UML de Clases

En el siguiente diagrama se muestran las clases más relevantes de la aplicación con las principales variables y métodos que implementa cada una, así como la relaciones entre ellas.

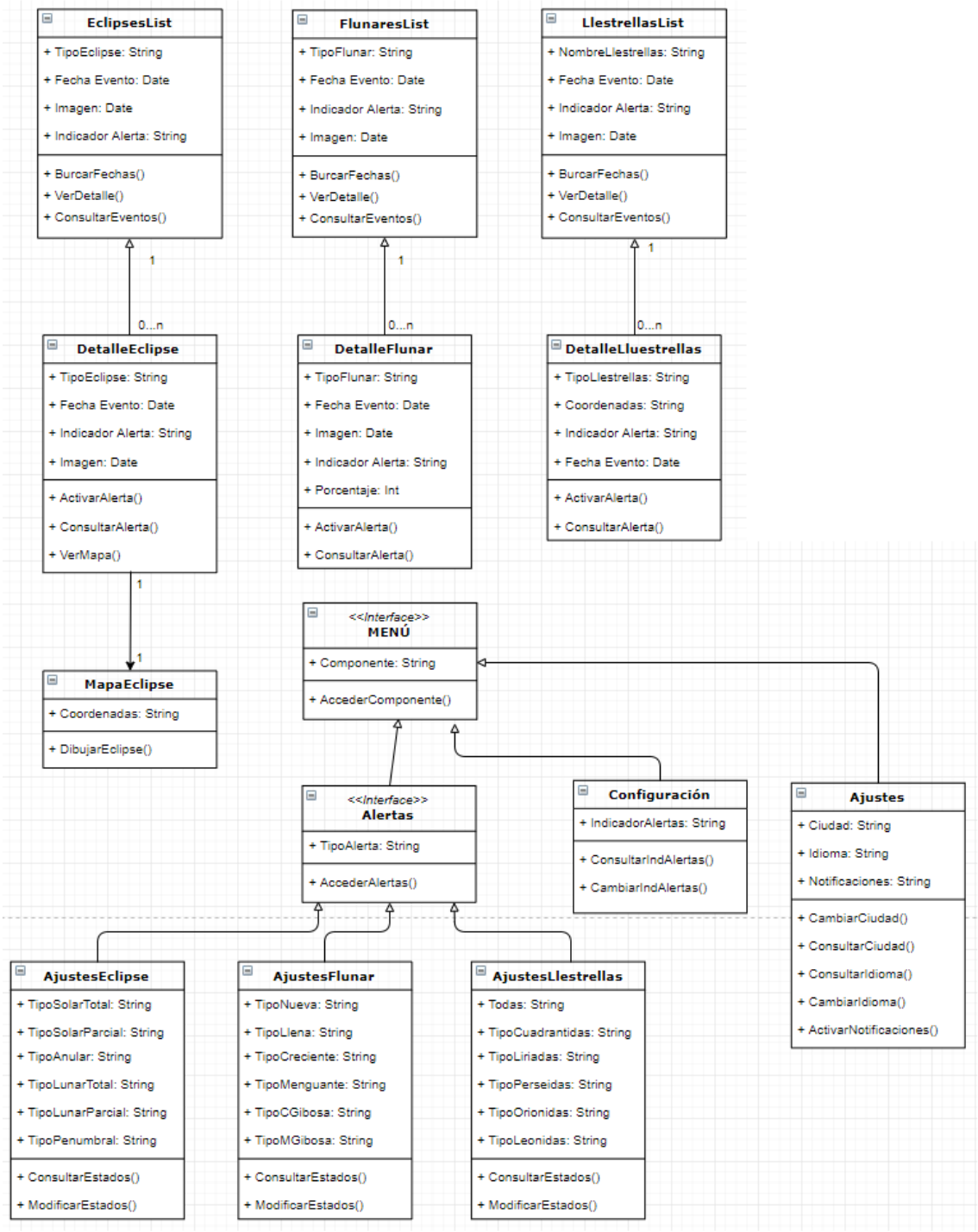


Figura 19. Diagrama de Clases

2.6.3. Diagrama de arquitectura de sistema

El siguiente diagrama representa la arquitectura de la aplicación. Se muestran dos ubicaciones de contenido diferenciadas, una en el propio dispositivo y otra en un servidor.
El acceso a servidor se hará a través de protocolo https.

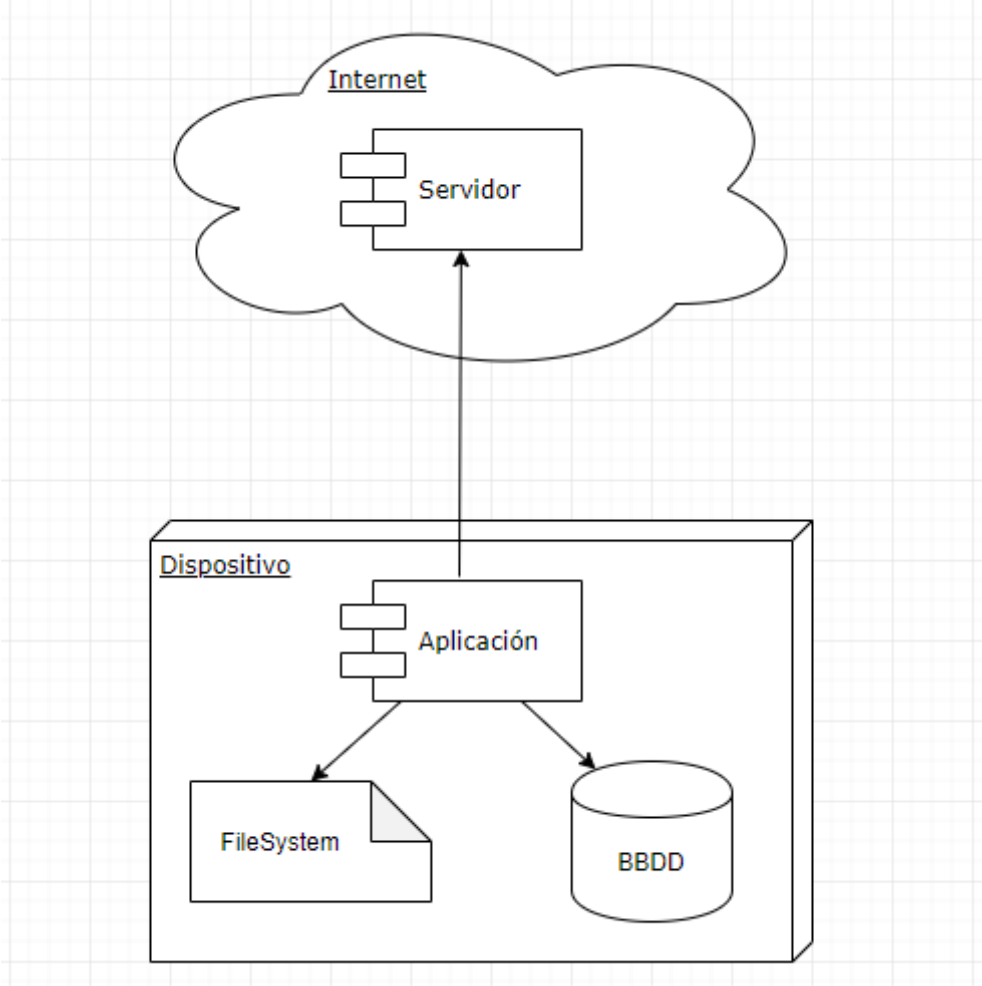


Figura 20. Diagrama de Arquitectura

3. Implementación

Es el momento de llevar a cabo el desarrollo del proyecto que hemos diseñado en los puntos anteriores. Como hemos comentado se trataría de desarrollar una aplicación móvil para Android
A continuación, vamos a estructurar este punto en varios apartados con el fin de explicar el contenido final del proyecto, los pasos realizados, las decisiones tomadas, etc.

3.1. Entorno de desarrollo

Se ha desarrollado el proyecto en el IDE oficial, Android Studio en su versión para Mac, concretamente estos son los datos de versión de IDE y SO.

Android Studio 3.2
Build #AI-181.5540.7.32.5014246, built on September 17, 2018
JRE: 1.8.0_152-release-1136-b06 x86_64
JVM: OpenJDK 64-Bit Server VM by JetBrains s.r.o
macOS 10.14.2

Requisitos del dispositivo o emulador:

minSdkVersion 26
targetSdkVersion 28
Conexión a internet

3.2. Estructura del proyecto

La estructura del proyecto respeta la propuesta por el IDE de Android Studio, donde los principales directorios que encontramos y con los que trabajaremos son:

- manifests: contiene el archivo AndroidManifest.xml.
- java: contiene los archivos de código fuente de Java, incluido el código de prueba JUnit.
- res: Contiene todos los recursos, como diseños XML, cadenas de IU e imágenes de mapa de bits.

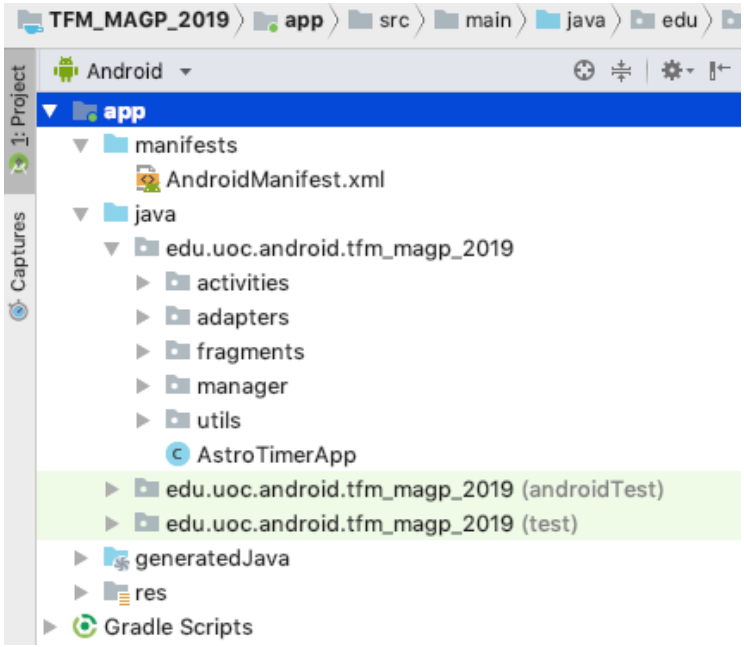


Figura 21. Estructura proyecto Android Studio

Respetando esta jerarquía de directorios hemos implementado todas nuestras clases java bajo el directorio java, paquetizando por tipo de clase o funcionalidad para mantener un orden.

activities: Contiene todas las *activities*, clases que extienden de *AppCompatActivity*.

adapters: contiene las clases encargadas del tratamiento de los datos para presentar listados de *listview* en pantalla.

fragments: contiene las clases que extienden de *Fragment*

manager: contiene los manejadores de las diferentes funcionalidades que necesite la aplicación hacer uso.

utils: contiene las clases utilizadas como objetos.

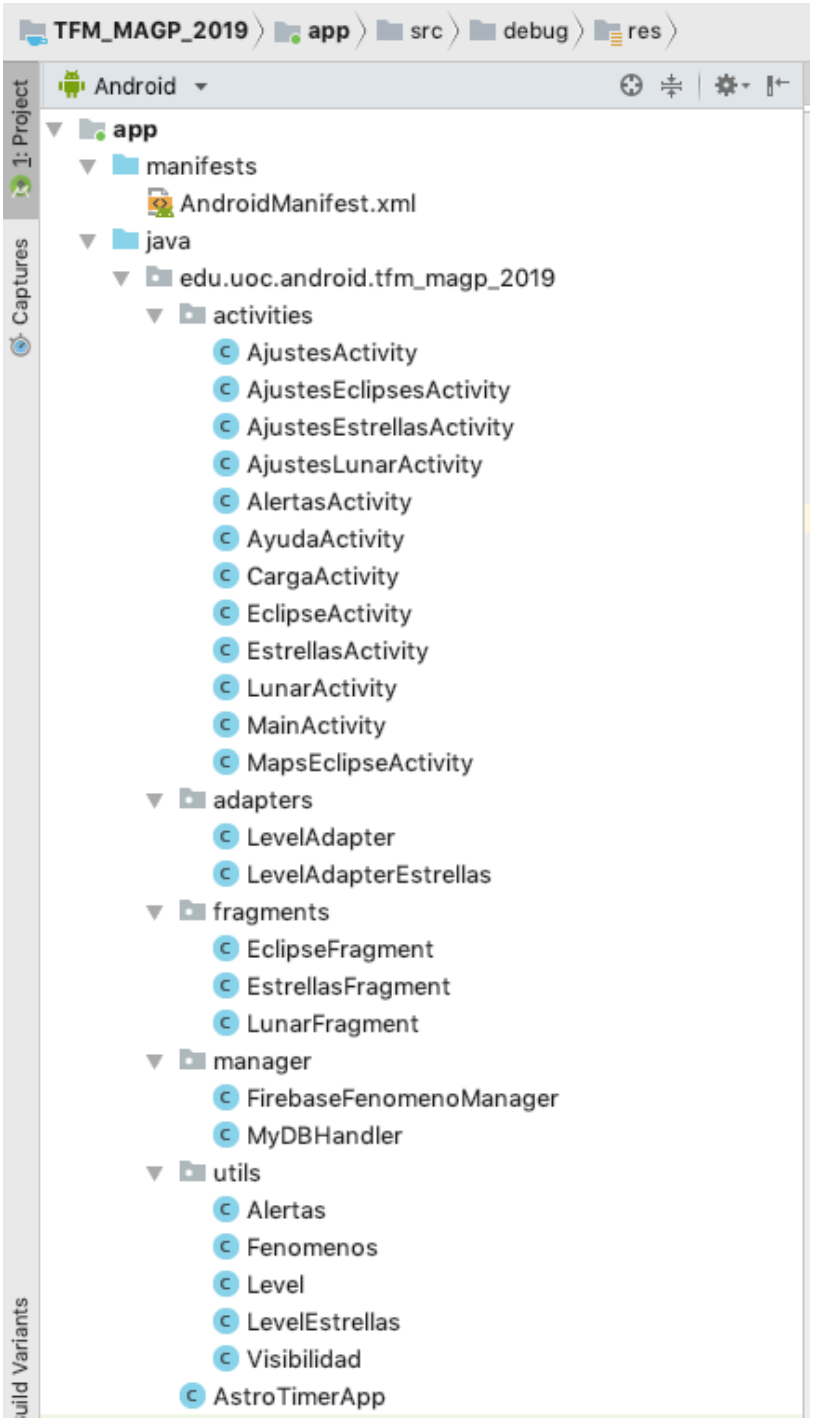


Figura 22. Estructura TFM 1

Dentro del directorio *res* de recursos, principalmente hemos trabajado sobre directorio *layout* donde están contenidos todos los ficheros XML de diseño de ventanas. Y hemos añadido en *drawable* las imágenes y ficheros necesarios para el diseño de la *app*.

Figura 23. Estructura TFM 2

3.3. Desarrollo

El primer paso en el desarrollo del proyecto, tras la preparación del entorno de desarrollo, ha sido la creación del propio proyecto al que hemos denominado AstroTimerApp. Vamos a comentar los elementos de desarrollo que hay detrás de cada flujo y funcionalidad de la aplicación.

Manifiesto

Es necesario añadir los permisos relacionados con la conexión a internet.

```
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
```

También es necesario indicar la clave de API de Google Maps para su uso.

```
<meta-data
    android:name="com.google.android.geo.API_KEY"
    android:value="@string/google_maps_key" />
```

Y cabe destacar la configuración necesaria para que la aplicación solo pueda ser usada en modo *portrait*

```
android:screenOrientation="portrait"
```

Carga de la aplicación

Comenzamos con el flujo de uso natural abriendo la aplicación, el flujo que esto desencadena es el siguiente.

AstroTimerApp: establece la conexión con el servidor *Firebase*[6] y habilita la persistencia de datos.
CargaActivity: Es la *activity* que se muestra en pantalla mientras se establece la conexión y los datos del servidor son descargados. También aquí se comprueba la conexión a internet necesaria para la descarga.

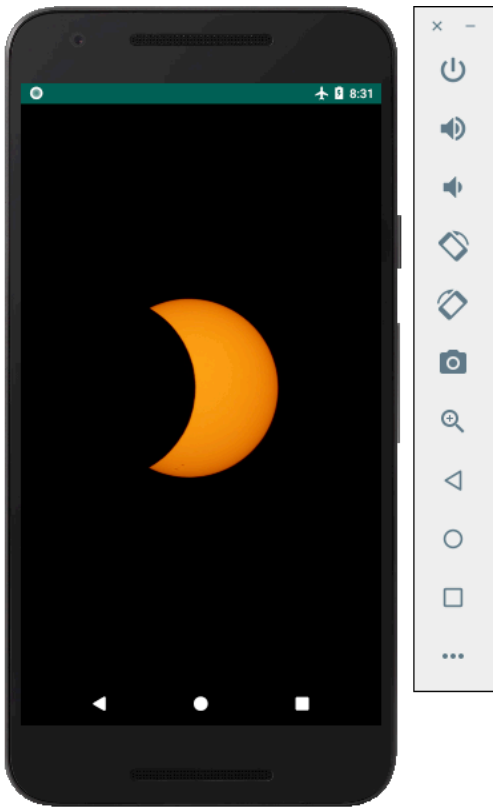


Figura 24. Vista de carga de aplicación

[6] Firebase: plataforma para el desarrollo de aplicaciones web y aplicaciones móviles desarrollada por James Tamplin y Andrew Lee en 2012 y adquirida por Google en 2014 <https://firebase.google.com>

FirebaseFenomenoManager: es el controlador desde donde se realiza la comunicación con el servidor y se obtienen los datos.

MainActivity: Una vez finalizado el proceso de carga cambiará el *activity* mostrado en pantalla y mostrará el menú principal. Esta *activity* se encarga básicamente de crear el menú, vinculando cada pestaña del *ViewPager* al *fragment* correspondiente.

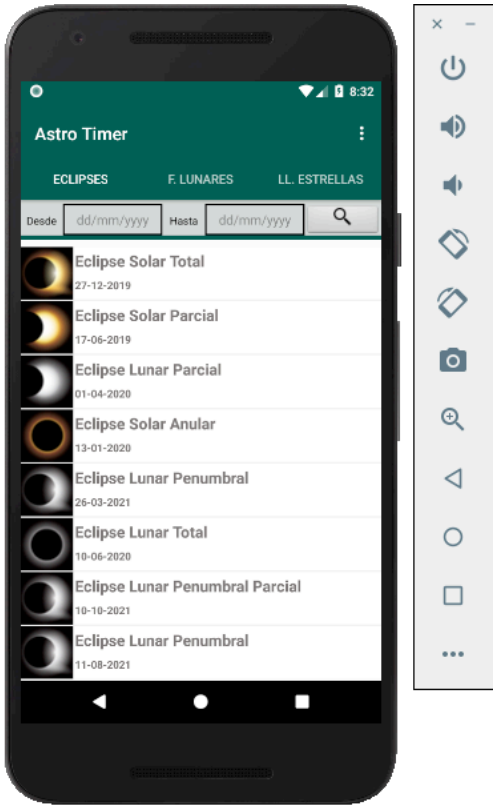


Figura 25. Vista pestaña Eclipses

Una vez en este punto habremos accedido al menú principal de la aplicación, desde el que tenemos acceso a las tres pestañas de datos de los diferentes fenómenos, así como a los menús de servicio.

Continuamos comentando pestaña por pestaña su contenido y funcionalidad.

Pestaña de Eclipses

Dentro de la pestaña de Eclipses nos encontraremos con un buscador en la parte superior, compuesto por un rango de fechas desde-hasta y un botón para aplicar el filtro, e inmediatamente a continuación se encuentra el listado de eclipses.

EclipseFragment: Es en esta clase donde se implementa la funcionalidad de la pestaña eclipses. Lo primero que hace es recuperar los elementos del servidor y filtrar los que se corresponden con la tipología de eclipse para mostrarlos en un *ListView.* Cualquier elemento de esta lista puede ser

seleccionado y esta clase implementa la funcionalidad para abrir la *activity* de detalle de eclipses ***EclipseActivity*** y pasarle la información que necesita.
Los campos *EditText* de entrada usados para las fechas implementan un *DatePickerDialog* que muestra un calendario donde seleccionar una fecha.
La funcionalidad del botón de filtrado se encarga de realizar las validaciones de las fechas, así como de recargar la vista para mostrar los resultados correspondientes.
La carga de la lista es tarea de ***LevelAdapter*** que recibe un *array* con los datos a cargar y este se encarga de añadir cada componente a la lista.

EclipseActivity es la vista donde se mostrará el detalle de cada evento seleccionado de la lista. Esta clase recoge los datos recibe de ***EclipseFragment*** y los dispone en la pantalla además calcula una cuenta atrás hasta la fecha del evento. Implementa dos funcionalidades añadidas:

- La posibilidad de guardar una alerta para ese evento. Esta información la haremos persistir en la aplicación implementando una tabla en una *bbdd SQLite*[7].
- Y un botón para visualizar el punto optimo de visibilidad del evento en un mapa. Este botón enlaza con una nueva vista ***MapsEclipseActivity*** donde se ha implementado el API de Google Maps para poder mostrar un mapa y dibujar las coordenadas requeridas.

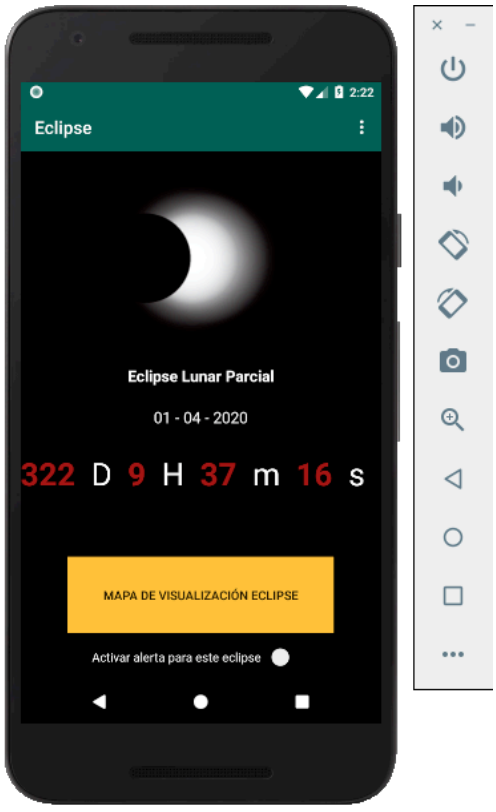


Figura 26. Vista detalle de Eclipses

[7] SQLite es un sistema de gestión de bases de datos relacional compatible con ACID, contenida en una relativamente pequeña biblioteca escrita en C. SQLite es un proyecto de dominio público creado por D. Richard Hipp. - *Wikipedia*

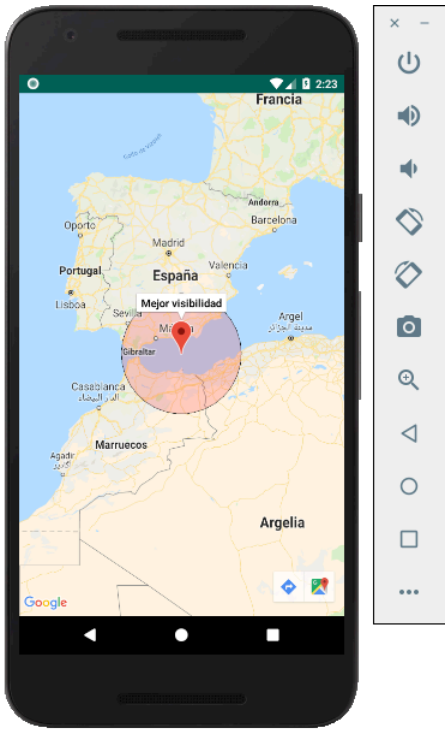


Figura 27. Vista mapa visualización Eclipse

Pestaña de Fases Lunares

El diseño de esta vista consta de un cuadro de texto de mandos superior donde se mostrará información relativa a la fecha que se está consultado y un buscador para cambiar la fecha y por tanto la información a mostrar.

Bajo este panel se encuentra la información de las fases lunares en formato de tabla, imitando un calendario gregoriano, donde cada celda dará acceso al detalle concreto de ese evento.

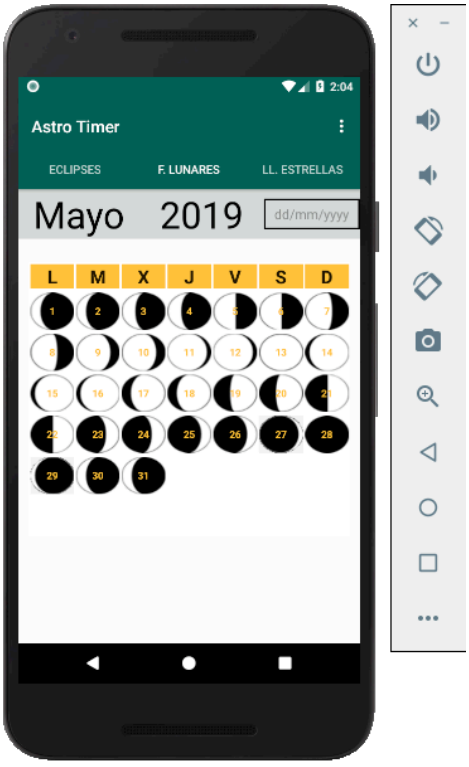


Figura 28. Vista pestaña F.Lunares

LunarFragment: Es la clase encargada de montar la vista de la pestaña F.Lunares. A partir de una fecha dada es capaz de mostrar el calendario correspondiente a ese mes. En este calendario se muestra en cada celda el día del mes y una imagen representativa del evento.
Cada una de estas celdas puede ser accedida y ejecuta una nueva *activity*, ***LunarActivity*** que se encarga de mostrar los datos correspondientes al evento seleccionado. Estos datos son enviados desde la *activity LunarFragment*

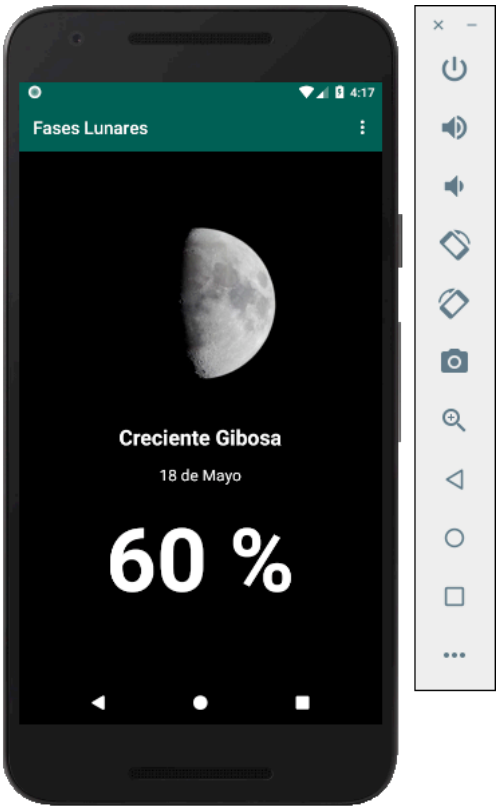


Figura 29. Vista detalle F.Lunares

Pestaña de Lluvia de estrellas

Esta pestaña muestra el listado completo de los eventos de tipo Lluvia de Estrellas.
Cada registro de la lista muestra una disposición similar a la de los ítems del listado de eclipses, con los mismos elementos, una línea para el nombre del evento y otra para la fecha, pero en este caso la imagen se muestra en una tercera línea.

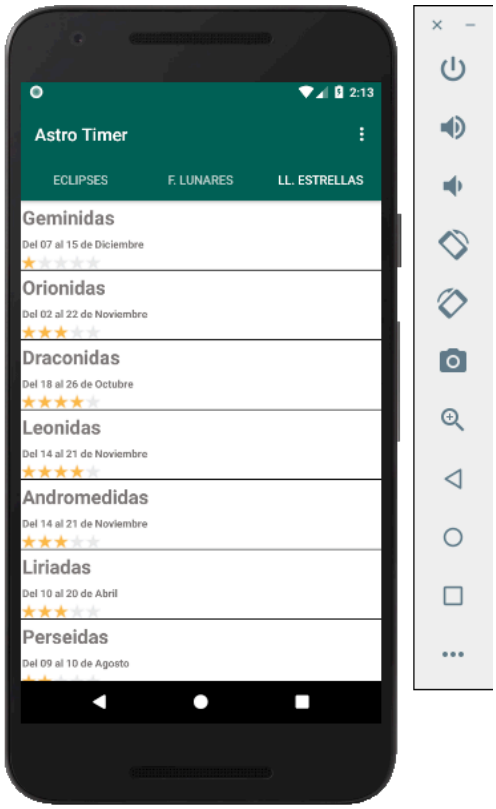


Figura 30. Vista pestaña LI.Estrellas

EstrellasFragment: Realiza la llamada a ***FirebaseFenomenoManager*** para recuperar todos los eventos descargados del servidor y filtra los que son de tipo Lluvia de Estrellas y guarda el resultado en un *ArrayList* que pasa a ***LevelAdapterEstrellas*** que se encarga de acceder a la información de cada registro y disponerla en la vista.

EstrellasActivity: Es la vista accedida por cada uno de los elementos de la lista anterior. Esta clase recibe información del evento seleccionado a través de *Intent*. La información que muestra es el nombre del evento, la fecha y como particularidad muestra una gráfica donde se representa la visibilidad del fenómeno en el rango de días que tiene lugar. Esta funcionalidad la hemos implementado usando la clase *LineChartData*, una clase que nos proporciona los métodos necesarios para la representación de gráficos.
También disponemos de la funcionalidad de activación de alerta que estará disponible en forma de botón *switch* que nos indicará si está o no activa.
Esta funcionalidad, compartida con eclipses, utiliza diferentes métodos de *MyDBHandler* para la consulta e inserción de la información en *bbdd*. Así puede consultar al entrar en la ventana si ya tiene la alerta activa y así mostrarla, o cambiar el estado actualizando el registro en la tabla tras modificar el botón *switch*.

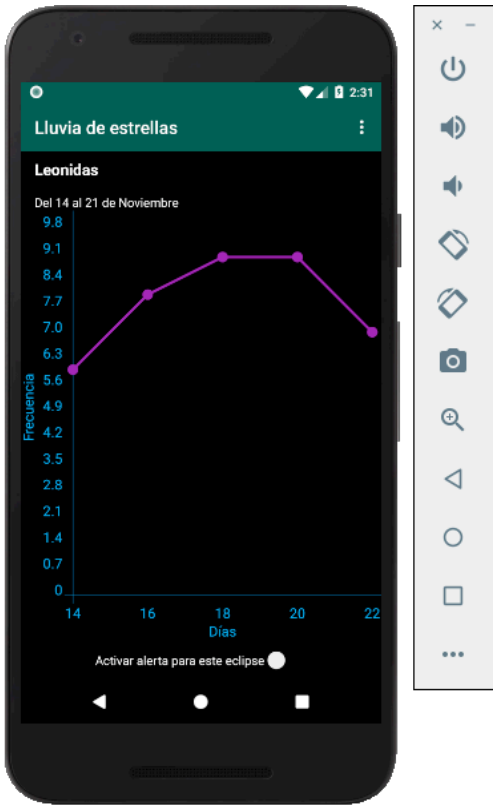


Figura 31. Vista detalle LI.Estrellas

Menú de servicio

Este menú está situado en la parte superior derecha del *ToolBar* y está implementado para ser usado desde todas las pestañas y ventanas de detalle de eventos. Es implementado por el método *onCreateOptionsMenu* y dispone de 3 opciones, ajustes, ayuda y alertas. Cada una de estas opciones tiene un *listener* que tras pulsar en una de ellas desencadena una llamada a su respectiva *activity*.

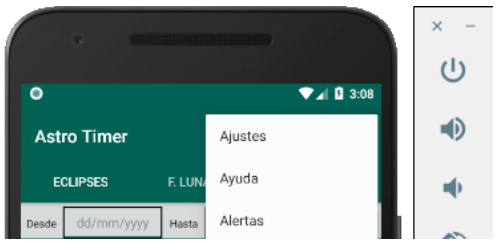


Figura 32. Vista menú de servicio

Opción Ajustes

AjustesActivity: Esta vista dispone de diferentes funcionalidades que permiten realizar acciones concretas sobre la aplicación, como son el cambio de ciudad, de idioma o la desactivación de todas las alertas.

En esta versión de la aplicación no se ha desarrollado funcionalidad para las opciones de Ciudad e Idioma.

La funcionalidad de borrado hace un *delete* de todos los registros de la tabla *alertas*, y así elimina todas las alertas configuradas.

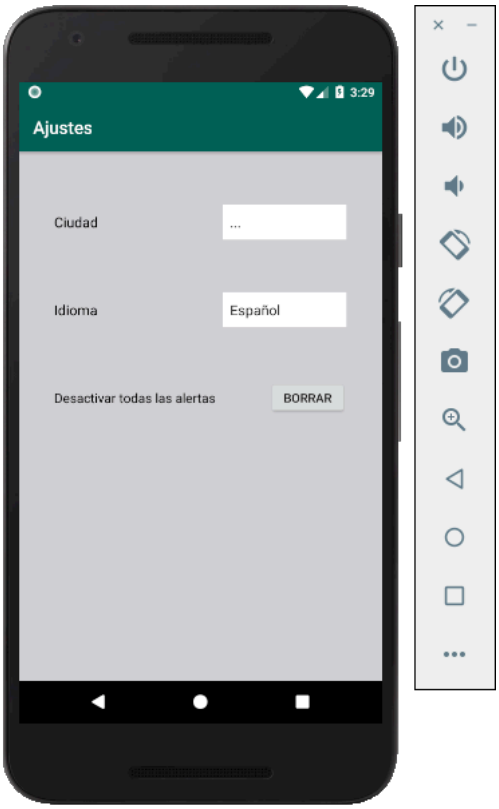


Figura 33. Vista ajustes

Opción Ayuda

AyudaActivity: Esta vista dispondrá diferentes cuadros de texto con guías de uso y resolución de dudas frecuentes.

Figura 34. Vista ayuda

Opción Alertas

Esta vista da opción a acceder a la configuración de alertas de los diferentes tipos de eventos, eclipses, fases lunares y lluvias de estrellas.

AjustesEclipsesActivity y AjustesEstrellasActivity: desarrollan la funcionalidad de activación y desactivación de alertas por tipo, así puedes activar o desactivar en una única acción todos los fenómenos de un tipo. Estas clases recupera la información guardada en *bbdd* para mostrar las opciones que están activas.

Para activar todas las alertas correspondientes a un evento se recupera la información descargada del servidor y se filtra por tipo y subtipo de evento para recuperar los códigos de estos, y esta información la que se hará persistir en la tabla de *alertas*

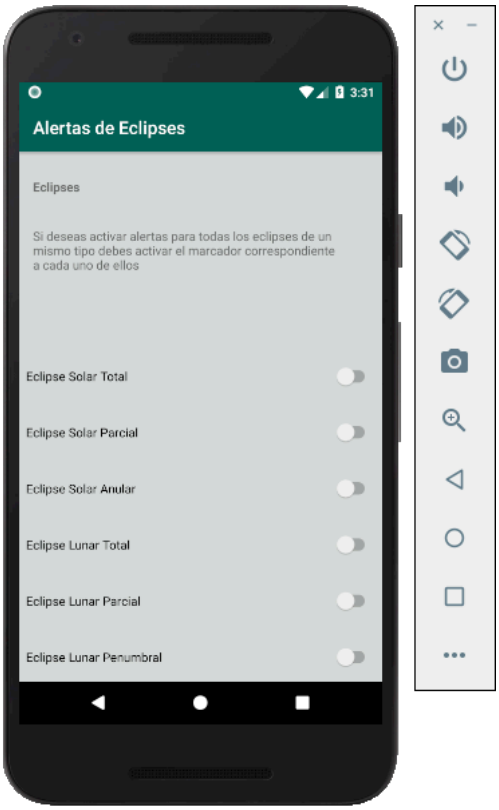


Figura 35. Vista alertas eclipse

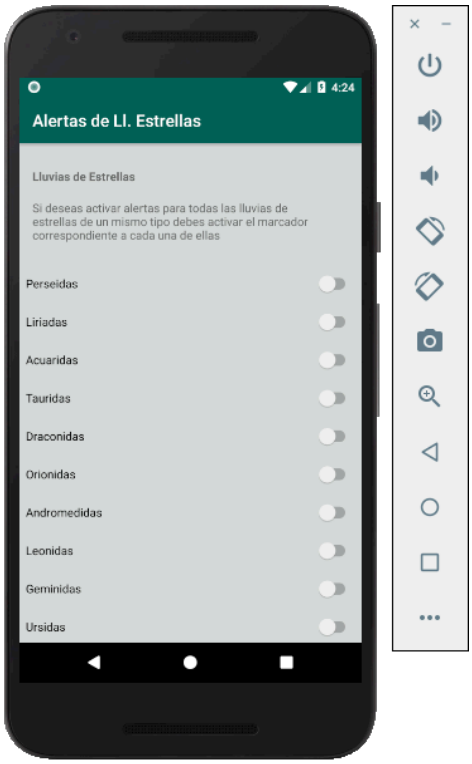


Figura 36. Vista alertas LI.Estrellas

3.4. Servidor

Para la parte servidor de la aplicación hemos aprovechado la integración que existe entre *Firebase* y Android Studio, y que esta nos ofrece los requisitos que necesitamos, para decantarnos por esta plataforma de *Back End* en servidor.

El requisito necesario en la parte servidor definimos que era poder acceder a información de los eventos que va a mostrar la aplicación en tiempo real. Esto lo hemos conseguido utilizando una base de datos en tiempo real (Realtime Database).

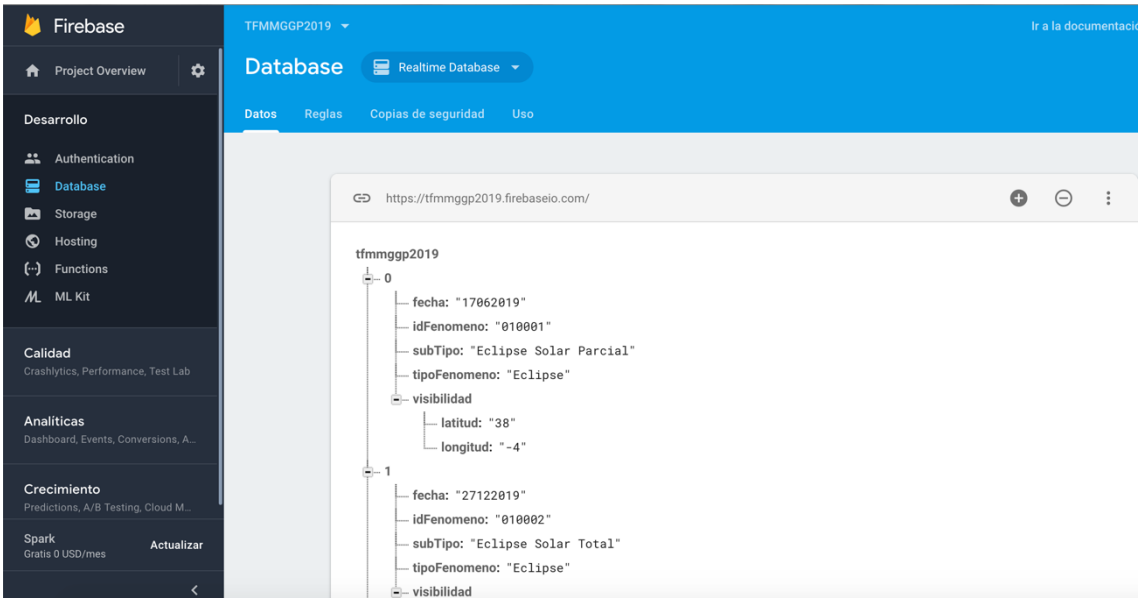


Figura 37. Firebase – captura de database

Ha sido necesario crear un nuevo proyecto en la consola de *Firebase* y asignar a este nuestro proyecto Android, una vez confirmada la conexión entre el proyecto y la consola tendremos acceso a sus funcionalidades. Y es aquí donde hemos subido el fichero *.json* en el que tenemos los datos que consumirá nuestra aplicación.

```
[
      {
      "tipoFenomeno":"Eclipse",
      "idFenomeno":"010001",
      "subTipo":"Eclipse Solar Parcial",
      "fecha":"17062019",
      "visibilidad":
            {
            "latitud":"38",
            "longitud":"-4"
            }
      },
      {
      "tipoFenomeno":"Eclipse",
      "idFenomeno":"010002",
      "subTipo":"Eclipse Solar Total",
      "fecha":"27122019",
      "visibilidad":
            {
            "latitud":"36",
            "longitud":"-3"
            }
      },
      …
```

Para el acceso a los datos desde nuestro proyecto *Firebase* nos proporciona una URL de acceso a la información, en nuestro caso:

https://tfmmggp2019.firebaseio.com/

Para poder hacer uso de esta información desde la aplicación, a parte de los pasos de sincronización y las librerías necesarias en Android Studio, deberemos dar permiso de lectura/escritura sobre los datos.

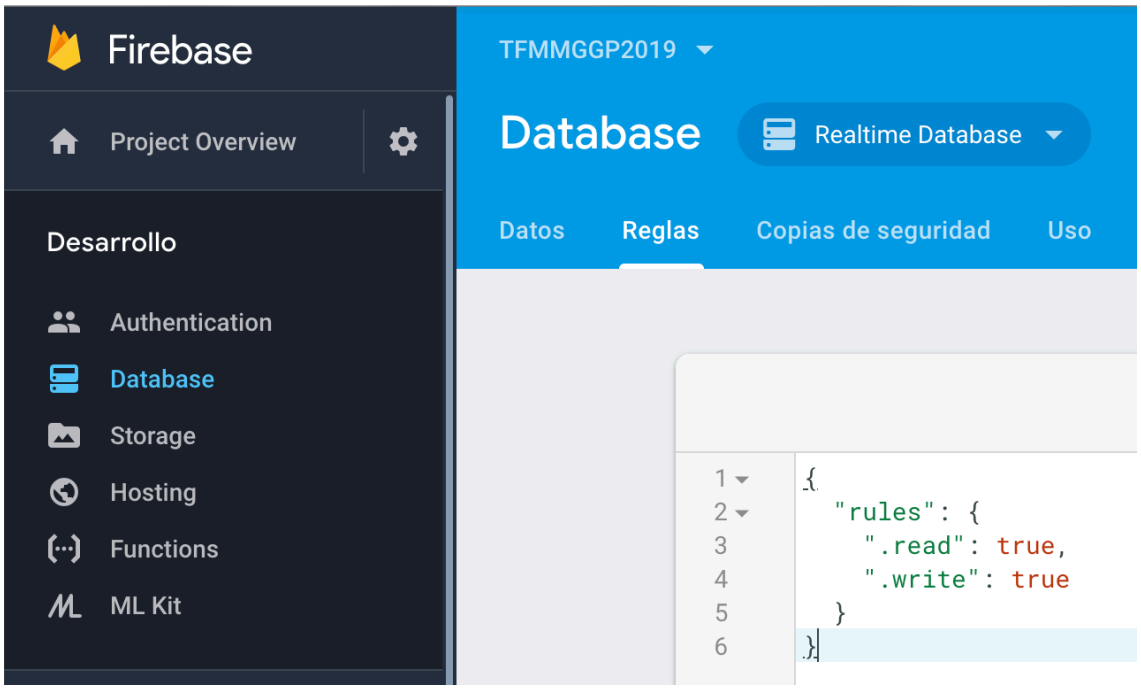


Figura 38. Firebase – permisos de acceso a datos

En este punto ya estemos en disposición de hacer uso desde la aplicación de la conectividad con nuestra base de datos online en *RealTime.*

3.5. SQLite

El sistema de almacenamiento en base de datos local lo hemos implementado con la API de *SQLite* que tenemos disponible en Android Studio.

Para el uso de las funcionalidades de *SQLite* necesitamos crear una clase que extienda de *SQLiteOpenHelper* y que contenga los métodos necesarios para nuestro tratamiento de datos. En nuestro caso esta clase es ***MyDBHandler*** que recoge los métodos de creación de las tablas de nuestra *bbdd*

```
@Override
public void onCreate(SQLiteDatabase db) {
    String CREATE_ALERTAS_TABLE = "CREATE TABLE " +
            TABLE_ALERTAS + "(" + COLUMN_ID + " INTEGER PRIMARY KEY," +
COLUMN_NAME
            + " TEXT " + ")";
    db.execSQL(CREATE_ALERTAS_TABLE);

}
```

así como de los métodos propios de manejo de datos, como pueden ser *addHandler*, *findHandlerId o deleteHandlerId.*

Un ejemplo de los datos guardados en la tabla de alertas visualizado desde la herramienta *DB Browser for SQLite*[8], donde podemos ver las columnas de la tabla y su contenido.

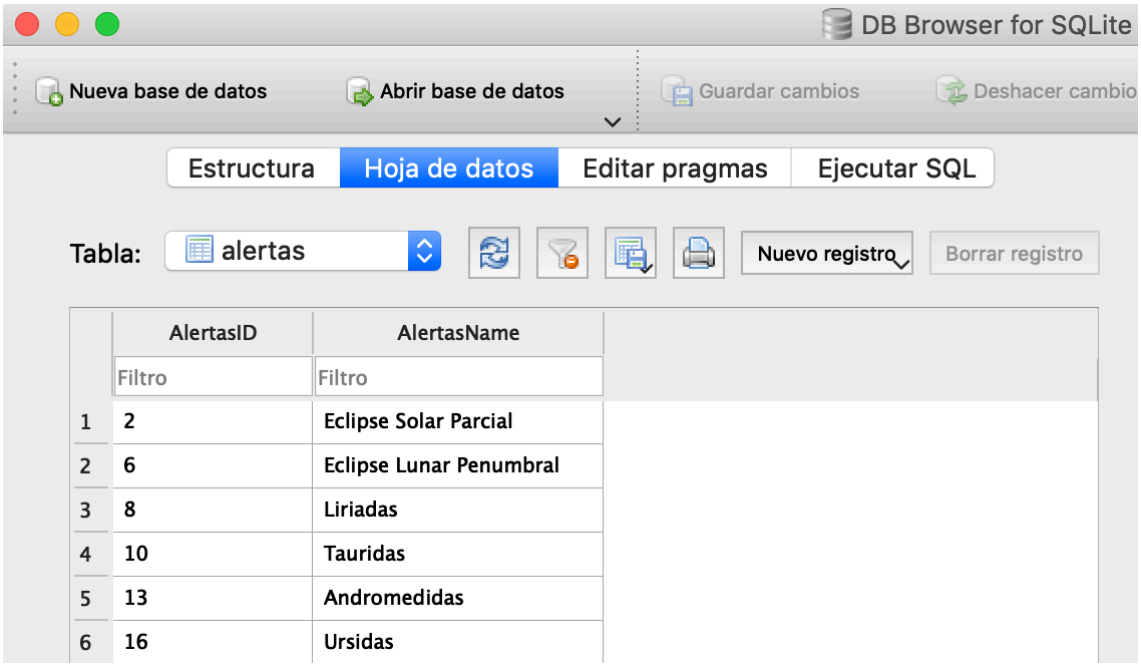


Figura 39. SQLite – datos de tabla alertas

4. Conclusiones

El producto es cambiante. En todo el ciclo de vida del proyecto, desde que se idea, cuando pasa por diseño, cuando se implementa, etc, es susceptible de mejora y evolución. Hay aspectos, partes de requisitos que de una fase a otra puede verse la necesidad de cambio, por requerimientos técnicos, funcionales, de diseño, y que, aunque se dedique todo el esfuerzo a prever cada escenario muchas veces no se es consciente de la problemática hasta que se comienza a desarrollar.

El equipo es importante. Para realizar la planificación de un proyecto de este estilo es necesario analizar muy bien de antemano que fortalezas y que debilidades tiene cada miembro del equipo, y más en este caso donde el equipo está formado por una única persona.

4.1. Seguimiento de la planificación

Diseño

En términos generales nos hemos ajustado en la ejecución de cada punto según lo planificado.

[8] *DB Browser for SQLite* es una herramienta visual, de código abierto para crear, diseñar y editar archivos de base de datos compatibles con SQLite

Ha sido en el *Diseño de la Arquitectura* donde hemos cambiado el paso respecto a lo planificado ya que habíamos dispuesto ejecutar cada apartado por separado en un total de 8 días repartiendo el tiempo por igual. Pues bien, a la hora de realizar los trabajos encontramos más práctico diseñar al mismo tiempo los diagramas de BBDD, clases y arquitectura por las sinergias existentes entre ellos y al trabajar así nos repercutía en mayor agilidad ante cambios.

Implementación

La planificación en este apartado hemos encontrado que, aunque realista, fue demasiado optimista. Nos hemos encontrado con problemas que han hecho que la planificación ideada sufriera cambios.

El principal problema ha sido el alcance del proyecto, ya que este no estaba dimensionado correctamente para ejecutar en los tiempos necesarios y con los recursos de que se dispone. Esto nos ha llevado a tomar la decisión de o bien replanificar el proyecto y proponer otra fecha de entrega, o asumir el sobreesfuerzo y repercutir más horas a cada apartado. Ponderando cuál sería el sobresfuerzo a realizar y estimándolo en un 40% (un total de 140%) decidimos asumirlo y respetar la fecha de entrega.

El aspecto que no valoramos adecuadamente a la hora de hacer la planificación ha sido el tiempo medio por línea de código que podría ejecutar, habiendo penalizado la falta de experiencia en el entorno.

Por tanto, el resultado ha sido que hemos tenido que dedicar un 40% más de esfuerzo del planificado durante el desarrollo del proyecto, imputando a partes iguales en desarrollo (20%) y pruebas (20%)

5. Glosario

activity: cada una de las pantallas o vistas que forman una aplicación.
agile: metodología de desarrollo
android: sistema operativo de Google
apk: extensión de fichero ejecutable de una aplicación Android
app/apps: acrónimo de aplicación
array: vector de una dimensión que almacena variables de un mismo tipo
arraylist: vector de una dimensión que almacena variables de un mismo tipo sobre el que se pueden añadir o eliminar registros.
back end: capa de acceso a datos
bbdd: siglas de base de datos
check: elemento que señala la activación o desactivación
core: funcionalidad principal o más importante
delete: sentencia de borrado en lenguaje de base de datos
drawable: recurso gráfico que se puede mostrar en pantalla
edittext: campo de texto de entrada
facebook: red social
feedback: retorno de apreciaciones
fragments: representa un comportamiento o una parte de la interfaz de usuario en una Activity.
google play: tienda de aplicaciones de google
google: compañía estadounidense que provee servicio de OTT
intent: clase que permite realizar peticiones en tiempo de ejecución
ios: sistema operativo de Apple
json: formato de texto sencillo para el intercambio de datos
label: etiqueta que contiene un texto
layout: define la estructura visual para una interfaz de usuario
listener: funcionalidad encargada de detectar interacciones sobre un elemento.
listview: colección de vistas de desplazamiento vertical, donde cada vista se coloca inmediatamente debajo de la vista anterior en la lista
mock: fotomontaje que permite diseñar una aplicación a alto detalle
OTT: acrónimo de over the top
portrait: orientación horizontal de la pantalla del dispositivo.
smartphone: ordenador de bolsillo con capacidades de teléfono móvil
sprints: es un marco de tiempo fijo repetible durante el cual se crea un producto
switch: widget que simula un pulsador, puede estar activo o desactivo
tablet: dispositivo móvil con capacidades de ordenar y tamaño intermedio entre el móvil y el ordenador.
toolbar: elemento de navegación superior en aplicaciones Android
twitter: red social
widget: elemento que proporciona una funcionalidad a una aplicación
xml: extensión de fichero de lenguaje similar a HTML.

6. Bibliografía

[1] Vanessa Rosselló Villán (03 OCT 2018)[en línea] Madrid.[Consulta 10 Mar 2019] [Disponible en: https://www.iebschool.com/blog/que-son-metodologias-agiles-agile-scrum/]

[2] Wikipedia (17 feb 2019) [en línea] [Consulta: 10 Mar 2019] [Disponible en: https://es.wikipedia.org/wiki/Manifiesto_%C3%A1gil]

[3] Herramienta para creación de Mocks <https://balsamiq.com/wireframes/>

[4] Herramienta online para la creación de prototipos de aplicaciones <https://www.justinmind.com/>

[5] Enlace a fuente de las reglas heurísticas de Jakob Nielsen. http://www.braintive.com/10-reglas-heuristicas-de-usabilidad-de-jakob-nielsen/

[6] Firebase: plataforma para el desarrollo de aplicaciones web y aplicaciones móviles desarrollada por James Tamplin y Andrew Lee en 2012 y adquirida por Google en 2014 <https://firebase.google.com>

[7] SQLite es un sistema de gestión de bases de datos relacional compatible con ACID, contenida en una relativamente pequeña biblioteca escrita en C. SQLite es un proyecto de dominio público creado por D. Richard Hipp. - Wikipedia

[8] DB Browser for SQLite es una herramienta visual, de código abierto para crear, diseñar y editar archivos de base de datos compatibles con SQLite

Enlaces de referencia:

Web oficial de Android para desarrolladores <https://developer.android.com/?hl=es-419>

Foro de desarrolladores para consulta de problemas, errores, dudas <https://es.stackoverflow.com/>

Consulta de proyectos <https://github.com/>

7. Anexos

Anexo 1. Entrevista cerrada a usuario

- Sí hablamos de fenómenos astronómicos como los eclipses, lluvias de estrellas, etc. ¿Cómo te definirías? Indiferente, curiosa, aficionada o experta.

Podría decirse que curiosa, si tengo la oportunidad me gusta verlo y disfrutarlo, pero no llega al nivel de afición donde deje otros planes por estos.

- ¿Cómo sueles enterarte cuando hay un evento de este tipo por prensa, tv, amigos…?

Pues por lo general, las veces que me entero, es por las noticias de tv.

- ¿Dispones de un Smartphone?

Sí, claro

- ¿Descargarías una aplicación en tu móvil para estar enterada de próximos eventos de este tipo?

Sí, creo que sí. Podría encontrarla interesante.

- Y respecto al contenido que esperas encontrar, preferirías una interfaz simple y minimalista (únicamente con la información necesaria) o verías más útil que muestre gran cantidad de información.

Pues la verdad, si me presenta una pantalla llena de datos que seguramente no entienda y no me aporten valor me resultaría incómodo. Prefiero que sea más simple, con información que me resulte útil para saber cuándo se producirá el evento y de que tipo será.

- Continuando con el contenido, ¿sería más útil para ti tener listada toda la información independientemente del tipo o consideras más adecuada ordenarla en diferentes pestañas o menús?

Probablemente sea más cómodo poder ver los eventos ordenados por tipo cada uno en su pestaña, será más sencillo encontrar la información que busque en ese momento.

- ¿Podrías ordenarme por preferencia estos aspectos de la aplicación? Interfaz atractiva, sistema de notificaciones, configuración por eventos (mostrar únicamente los tipos de evento que el usuario quiera)

A ver, en primer lugar, pondría el sistema de notificaciones ya que creo que es lo que me resultaría más práctico. A continuación, la configuración de eventos, puede que no me interese tener información de todos ellos y si pudiera configurarlos a mi gusto sería ideal. Por último, pongo la interfaz, tampoco

porque lo considere poco importante, sino porque lo considero un aspecto básico que se debe tener en cuenta siempre.

- ¿Crees que te resultaría útil un sistema de notificaciones?

Sí, como te he dicho antes, es la funcionalidad que considero más impórtate para esta aplicación. Si estas se pudieran llegar a configurar por tipo de evento e incluso por cada evento concreto sería ideal.

- Si llegado el momento, el sistema viera que no has programado ninguna notificación en un tiempo determinado, ¿te resultaría molesto recibir un aviso de aplicación para recordarte que puedes hacerlo?

Pues depende, hay aplicaciones que resultan bastante pesadas mostrando notificaciones cada poco para que vuelvas a usarlas. Por lo tanto, si no es muy invasivo lo vería bien.

- Si la aplicación te diera la opción de publicar información (WhatsApp, Twitter, Facebook…) de evento concreto, ¿la usarías?

La verdad, creo que no le daría mucho uso, quizás en algún evento puntual… pero por lo general creo que usaría la aplicación para recibir información, no para enviarla o compartirla.

- ¿En qué dispositivo instalarías esta aplicación, en un smartphone o en una Tablet?

En principio en el smartphone. La Tablet no la uso a diario y no hago mucho caso a las notificaciones.

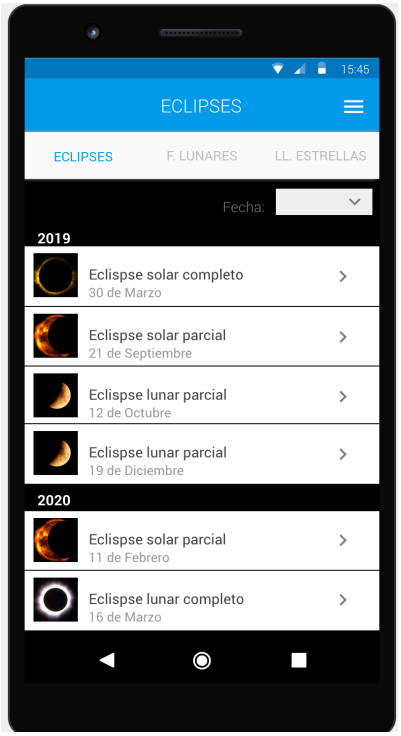
- Sí existieran dos opciones de descarga de esta app, una de pago y otra con publicidad, ¿Cuál elegirías? En caso de ser de pago, que cantidad consideras adecuada.

Pues yo creo que elegiría la versión con publicidad. No me pasaría mucho tiempo de la aplicación más que el necesario para programar las alertas por lo que no me resultaría molesto tener que lidiar con un poco de publicidad. En el caso de que me planteara comprarla, no creo que pagara más de 1€, creo que sería este un valor justo.

- ¿Consideras importante algún aspecto del que no hayamos hablado?

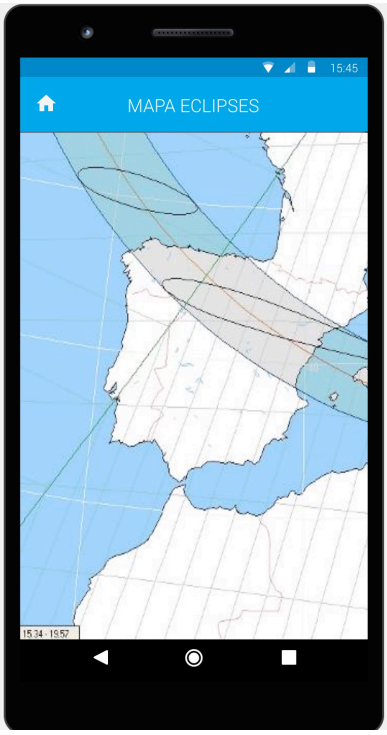
Pues déjame que piense… quizás, como muchos de estos eventos son cíclicos, estaría bien que pudiese apuntar notas sobre el evento para poder consultarlas la próxima vez e incluso adjuntar fotos. No se me ocurre nada más

Anexo 2. Capturas prototipo

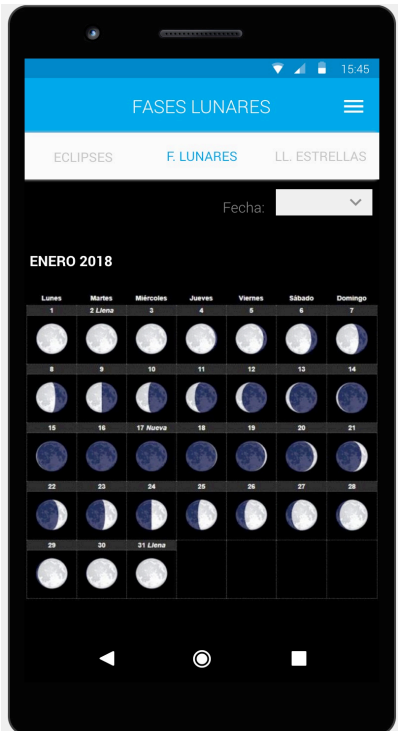


P.AT.01.00. Ventana de listado de Eclipses

P.AT.01.01. Ventana de detalle de Eclipse

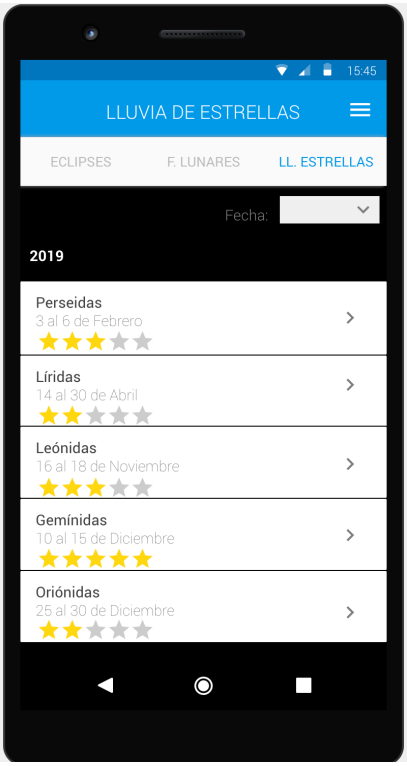


P.AT.01.02. Ventana de mapa de visualización de Eclipse

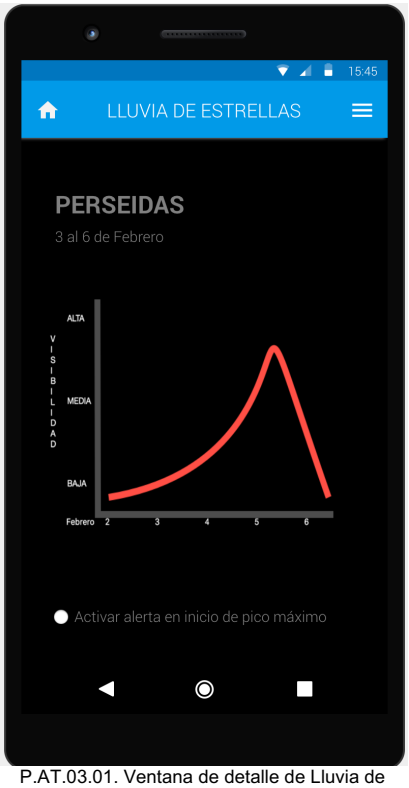


P.AT.02.00. Ventana de listado de Fases Lunares

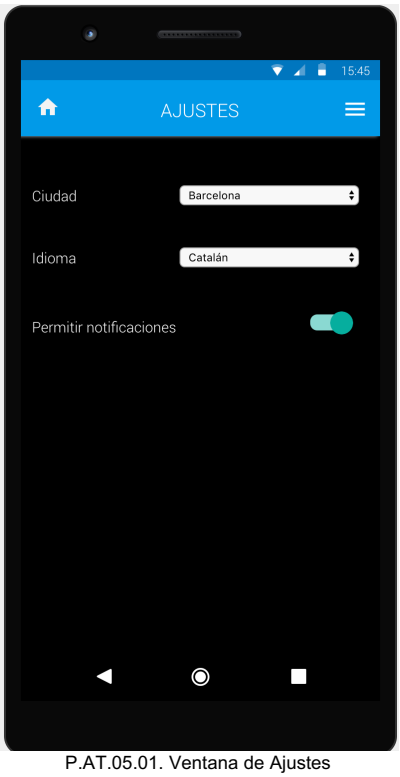
P.AT.02.01. Ventana de detalle de Fase Lunar



P.AT.03.00. Ventana de listado de Lluvias de Estrellas

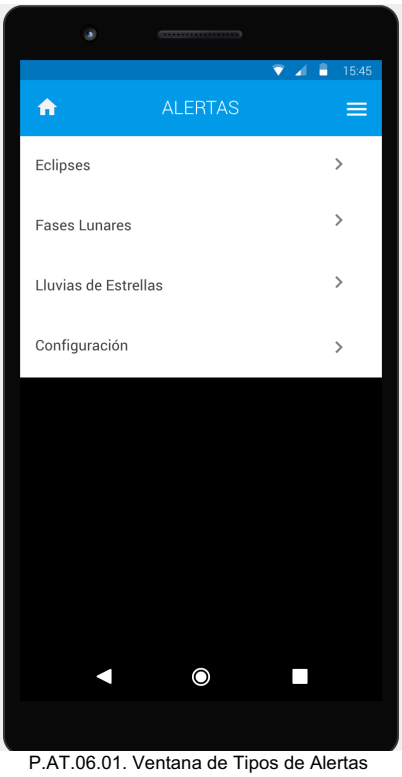


P.AT.03.01. Ventana de detalle de Lluvia de Estrellas



P.AT.05.01. Ventana de Ajustes

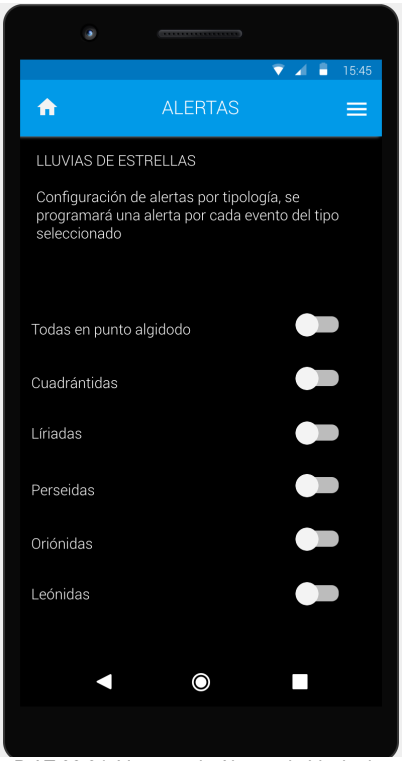
P.AT.04.01. Ventana de Ayuda



P.AT.06.01. Ventana de Tipos de Alertas

P.AT.06.02. Ventana de Alertas de Eclipses

P.AT.06.03. Ventana de Alertas de Fases Lunares



P.AT.06.04. Ventana de Alertas de Lluvia de Estrellas

P.AT.06.05. Ventana de configuraciones generales